VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2013 **Bc. Jan Svačina**

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Integrace VoiceXML do Asterisku

VoiceXML Integration into Asterisk

2013 **Bc. Jan Svačina**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Jan Svačina**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Integrace VoiceXML do Asterisku
VoiceXML Integration into Asterisk

Zásady pro vypracování:

Cílem diplomové práce je vytvoření hlasového menu v Asterisku s využitím Text to Speech (TTS). Zadání je součástí širší koncepce projektu, který zahrnuje vývoj systému distribuce hlasových zpráv, a proto je požadována interoperabilita s ostatními komponentami. Pro realizaci je nutností, aby na Asterisk byl provozován systém IVR (Interactive Voice Response). Pomocí VoiceXML na základě vstupů uživatele bude interpretován text ve formě řeči, což bude realizováno doplněním TTS modulu pomocí aplikace Festival.

1. VoiceXML, Asterisk, IVR a TTS.
2. Syntéza řeči a projekt Festival.
3. Návrh a integrace syntézy řeči do Asterisku s využitím VoiceXML.
4. Ověření a zhodnocení dosažených vysledků.

Seznam doporučené odborné literatury:

1. McGlashan, S., Burnet, D., Carter, J., Danielsen, P., Ferrans, J., Hunt, A., Lucas, B., Porter, B., Rehor, K., Tryphonas, S.: Voice Extensible Markup Language (VoiceXML) Version 2.0, W3C - Recommendation (2004).
2. Voznak, M., Rezac, F., Zdralek, J.: Danger Alert Communication System. In: IWSSIP 2010 - 17th International Conference on Systems, Signals and Image Processing, Rio de Janeiro, Brazil, June 17-19 2010, ISBN 978-85-228-0565-5.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 1.5. 2013

.....
podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Miroslavu Vozňákovi Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Tato diplomová práce se zabývá jazykem VoiceXML a jeho implementací do VoIP ústředny Asterisk. Jsou zde popsány jednotlivé možnosti jazyka, struktura VoiceXML dokumentů, jejich vytváření a interpretace pomocí VoiceXML interpreta VoiceGlue. Práce se dále zabývá zpracováním výstupu VoiceXML dokumentu, ten je proveden formou syntézy řeči za pomocí správce řečových syntéz Festival. V rámci diplomové práce byly vytvořeny dvě VoiceXML aplikace, které demonstrují schopnost spolupráce zadaných součástí. První aplikace vytváří informační dopravní servis pro české řidiče. Poskytuje jim aktuální informace o stavu nejfrekventovanějších silnic a dálnic v České republice a týdenní předpověď počasí. Druhá aplikace má za úkol číst aktuální informace ze serveru Novinky.cz nevidomým a zrakově postiženým lidem.

Klíčová slova

VoiceXML, VoiceGlue, Asterisk, Festival, převod textu na řeč, VoIP, IVR, PBX, ústředna

Abstract

This diploma dissertation is dealing with the VioceXML language and its implementation into Asterisk VoIP PBX. There are all possibilities of language, structure of VoiceXML documents, their creation and interpretation using VoiceXML interpreter VoiceGlue. The thesis focuses on the processing of the output VoiceXML document, which is conducted through speech synthesis by the Festival speech synthesis system. Two VoiceXML applications that demonstrate the ability to works given part together were created in this thesis. The first application creates a traffic information service for the Czech drivers. It provides them current information of the status on the busiest roads and motorways in the Czech Republic and weekly weather forecast. The second application is responsible for reading news from the server Novinky.cz to blind and visually impaired people.

Key words

VoiceXML, VoiceGlue, Asterisk, Festival, Text to Speech, VoIP, IVR, PBX, exchange

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
3G	Third Generation	Třetí generace
AGI	Asterisk Gateway Interface	Rozhraní brány Asterisku
API	Application Programming Interface	Rozhraní pro programování aplikací
DOM	Document Object Model	Objektový model dokumentu
DTMF	Dual-Tone Multi-Frequency	Tónová volba
GPL	General Public License	Všeobecná veřejná licence
HTML	HyperText Markup Language	Značkovací jazyk pro hypertext
HTTP	HyperText Transfer Protocol	Protokol výměny hypertextových dokumentů
IAX	Inter-Asterisk eXchange	Vnitřní protokol Asterisku
IM	Instant Messaging	Komunikační služba
IP	Internet Protocol	Internetový protkol
ISDN	Integrated Services Digital Network	Digitální síť integrovaných služeb
IVR	Interactive Voice Response	Interaktivní hlasový systém
PBX	Private Branch eXchange	Pobočková telefonní ústředna
PSTN	Public Switched Telephone Network	Veřejná telefonní síť
RFC	Request For Comment	Žádost o komentář
RSS	Really Simple Syndication	Souborový formát
SIP	Session Initiation Protocol	Protokol pro inicializaci relací
SMS	Short message service	Služba krátkých textových zpráv
TCP	Transmission Control Protocol	Komunikační protokol

Zkratka	Anglický význam	Český význam
TTS	Text To Speech	Převod textu na řeč
URL	Uniform Resource Locator	Jednotný lokátor zdrojů
VoIP	Voice over Internet Protocol	Internetová telefonie
VXML	VoiceXML	Jazyk VoiceXML
XML	Extensible Markup Language	Rozšířitelný značkovací jazyk

Obsah

1 Úvod

Člověk, jehož přirozeným způsobem komunikace je řeč, se snaží si co nejvíce zjednodušovat každodenní rutinní záležitosti, které zahrnují i veškerou komunikaci se světem. V dnešní moderní době není nijak složité komunikovat s kýmkoli, ať je kdekoli na světě. Díky internetu a jeho poměrně jednoduchému a levnému používání, se upřednostnila komunikace pomocí posílání různých zpráv ve formě e-mailů, IM zpráv, SMS a podobně, která ve většině případů dostačuje základním požadavkům a není nijak potřeba použití jiných prostředků pro výměnu informací. Pokud však chceme vést moderní dialog s dalšími osobami, existují technologie, které zajistí rychlé a kvalitní předání informací. Použitím těchto technologií již nemusíme být omezeni jen na psanou formu zpráv, ale můžeme předávat informace přirozenějším způsobem, řečí. Vést mluvený dialog mezi dvěma nebo i více osobami není vzhledem k vyspělé telekomunikační síti nikterak nemožné, ovšem při komunikaci s elektronikou jako jsou mobilní telefony, navigace nebo i domácí spotřebiče se neobejdeme bez ovládacích tlačítek nebo dálkových ovladačů. Pro zrychlení a zjednodušení moderního života je v podstatě nevyhnutelný přechod na ovládání těchto zařízení řečí, pro tento způsob komunikace je zapotřebí použití dialogových systémů.

Podobné systémy jsou již použity v komunikaci obchodních společností se zákazníkem, v kombinaci s telefonií vznikly takzvané hlasové portály. Komunikace pomocí takového portálu probíhá tak, že systém automaticky přijme zákazníkův hovor a přepojí do dialogového systému, kde umožní zákazníkovy výběr z nabídky stlačením příslušného tlačítka (DTMF volba). Dialogové systémy jsou tedy schopny převzít uživatelský vstup, zpracovat jej a podle vnitřního algoritmu vytvořit odpověď. Jako uživatelský vstup může být tónová volba, ale i samotný hlas a jako výstup k uživateli lze použít syntetizovanou řeč. Tyto systémy tedy dovedou vést dialog podle předem vytvořeného modelu, nebo se mohou podle dialogu adaptovat na danou situaci.

Cílem této práce je představení technologie VoiceXML a její implementace do pobočkové ústředny Asterisk

2 Dialogové systémy

Dialogové systémy umožňují komunikaci mezi uživatelem a počítačovým, nebo jiným automatizovaným systémem. Komunikace probíhá přirozenou formou, což je jejich hlavní výhoda. Tyto systémy nabízí mnoho modulů pro komunikaci s uživatelem, např. hlasové, textové, grafické a mnoho jiných. Forma vstupu od uživatele je vybírána pomocí různých otázek a samotná komunikace se systémem je pak vedena volně a většinou je i snadno ovladatelná. [1]

2.1 Historie

První systémy, které umožňovaly rozhovory s počítačem pomocí přirozeného jazyka začaly vznikat v 60. letech 20. století, a poskytovaly pouze textový vstup a výstup. Na začátku se jednalo o jednoduchý dialogový systém ELIZA, naprogramovaný v letech 1964 – 1966, jehož autorem byl Joseph Weizenbaum. ELIZA byl program schopný imitovat psychoterapeuta, vedl anglicky psaný dialog s uživatelem, a stal se jedním z milníků v komunikaci mezi člověkem a počítačem. Stal se také inspirací pro počítačové hry založené na dialogu.

Dalším zajímavým programem byl PARRY, ten sloužil pro simulaci neurotických procesů. Základem byla znalost a předpoklady psychologů, kteří pak po několika prvních simulacích nebyli schopni rozeznat program od skutečného paranoidního schizofrenika. V následujících letech pak mnoho nadšenců propadlo výzkumu těchto technologií, čímž přinesli mnoho poznatků o formování a zpracovávání lidského hlasu. S rozvojem vědy a techniky vznikaly nástroje, které kromě vědeckého uplatnění našly i uplatnění praktické. Jednalo se např. o systémy vyhledávání informací v přirozených jazycích, nástroje umožňující používání počítačů a jiných zařízení zrakově postiženým lidem apod. [2]

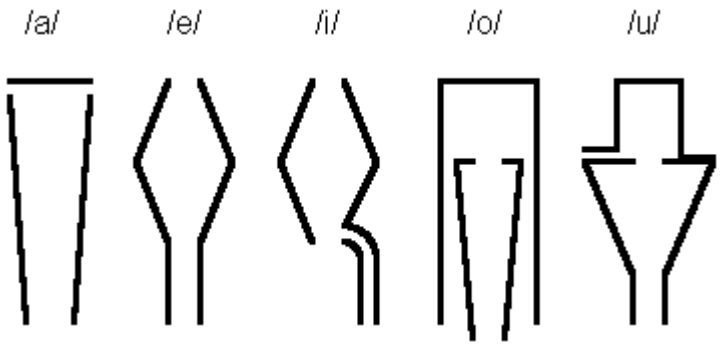
2.2 Zpracování lidské řeči

Proces zpracování lidské řeči zahrnuje dvě základní etapy. První je samotné porozumění řeči, kde je důležitá nejen schopnost převodu řeči na text, ale také samotné porozumění textu. Druhou etapou je vytvoření řeči, tzv. syntéza.

Rozpoznávání lidské řeči je komplexní záležitost, která vyžaduje použití složitých metod. Vývojem systému schopného porozumět sémantice lidské řeči se experti zabývají již mnoho desítek let, a čím více ve vývoji pokročí, tím častěji se potýkají s množstvím problémů a s omezeními dnešních programových i hardwarových komponent. Jeden z problémů se vyskytuje již při zaznamenávání hlasového vstupu, s nímž se zaznamenávají i různé ruchy a šumy. Odfiltrování takových šumů je poměrně složité, protože se musí rozhodnout co je relevantní vstup a co ne. Typický problém rozlišení šumu a sykavky, nejdříve se musí nahrávka rozdělit, musí se určit začátek a konec vstupu, pak se použijí různé analytické krátkodobé metody, rozdělení na časovou a frekvenční

doménu. Analyzovaný mikrosegment je pak testován na periodicitu, obecně však periodický není, a je potřeba použít váhové okénko, které určitým způsobem změní hodnoty segmentu. Analýza v časové doméně se zabývá hodnotami vzorku a předpokládá se, že na tomto časovém intervalu nedochází k výraznějším dynamickým změnám. V této analýze se používají různé průměrové funkce, funkce rozhodující lokální minima a maxima a další. Na rozdíl od toho analýza ve frekvenční doméně nejprve rozloží signál na základní komponenty, přičemž se nejčastěji používá funkce rozkladu spektra Fourierovou transformací. Obě složky jsou pak zapsány do akustického vektoru a ten je potom porovnáván s gramatickým předpisem.

Syntéza řeči je taktéž velmi složitý proces, u kterého je potřeba nejen vyslovit daný text, ale také zahrnout přenášené zosobněné vlastnosti řečníka. Již v minulosti se lidé snažili napodobovat lidský hlas pomocí mechanických pomůcek a různých nástrojů, např. rezonátory vytvořené ruským profesorem Christianem Kratzensteinem zobrazené na obrázku 2.1, které byly schopné modelovat samohlásky.

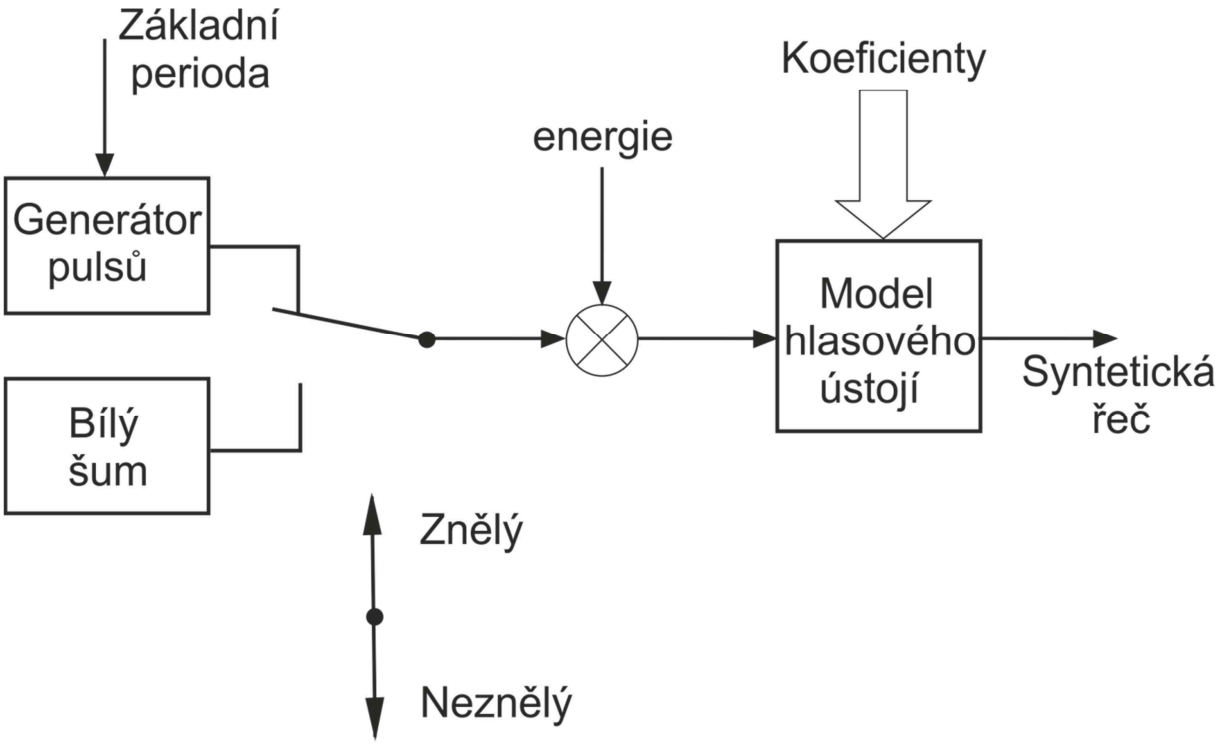


Obrázek 2.1: Kratzensteinovy rezonátory

Wolfgang von Kempelenův stroj dokázal nejen vyslovit samohlásky nebo souhlásky, ale také i celá slova. Poznatky z těchto let byly použity při tvorbě prvních počítačových systémů.

Podobně jako u analýzy signálu probíhá i u syntézy rozdělení na frekvenční a časovou oblast. K frekvenční syntéze se používá modelování lidského ústrojí pomocí frekvenčních generátorů, různých filtrů a zesilovačů. Časová syntéza pak vytváří slova, věty a větná spojení. Celá syntéza pak závisí na zvukových vlastnostech jazyka, které ovlivňují mnoho faktorů v globálním kontextu syntetizované řeči. [3]

Současné metody vytváření řeči se řídí jednoduchým modelem, viz obrázek 2.2, který je založen na modelu hlasového traktu, který napodobuje lidské hlasové ústrojí. Hlasový trakt lze realizovat číslicovým filtrem, který by měl být časově nestacionární.



Obrázek 2.2: Blokové schéma vytváření řeči

Buzení hlasového ústrojí závisí na tom, jestli je uvedený krátký segment znělý nebo neznělý. Jestliže se jedná o znělý segment, modelování kmitání hlasivek se provádí generátorem pulsů o základní hlasivkové periodě T_0. V případě neznělé části řeči se modelování provádí generátorem bílého šumu s plochým výkonovým spektrem. Skutečná řeč se ale neskládá vždy jen z čistě znělých nebo čistě neznělých segmentů, proto se v takovémto případě používá směšování obou těchto zdrojů. Pro lepší přiblížení skutečnosti lze ke znělým částem řeči přidat také generátor barevného šumu s definovanou plochou výkonového spektra. [4]

2.3 Součásti systému

Vzhledem k velkému množství dialogových systémů s různou architekturou, je obsah součástí těchto systémů různorodý, a nelze tedy přesně určit, jaké každý systém obsahuje. Mohou se řídit standardy jak je VoiceXML nebo mohou implementovat své vlastní návrhy. [1]

2.3.1 Dialogový manager

Je hlavní a obecně nejdůležitější součást dialogových systémů, spravuje stavy dialogu a jejich strategii, získává obsah uložený v souborech, databázích nebo jiných úložištích, dále spravuje dialogovou strategii a historii a rozhoduje o odezvě na základě uživatelského vstupu a stavu, ve kterém se dialog nachází.

2.3.2 Porozumění přirozeného jazyka

Tento proces je velmi složitý, protože je potřeba brát v úvahu různé rysy vstupu. Hlavní náplní komponenty je vytvořit syntaktické a sémantické schéma. Obsahuje jmennou identifikaci, značkování slovních druhů a syntaktickou analýzu.

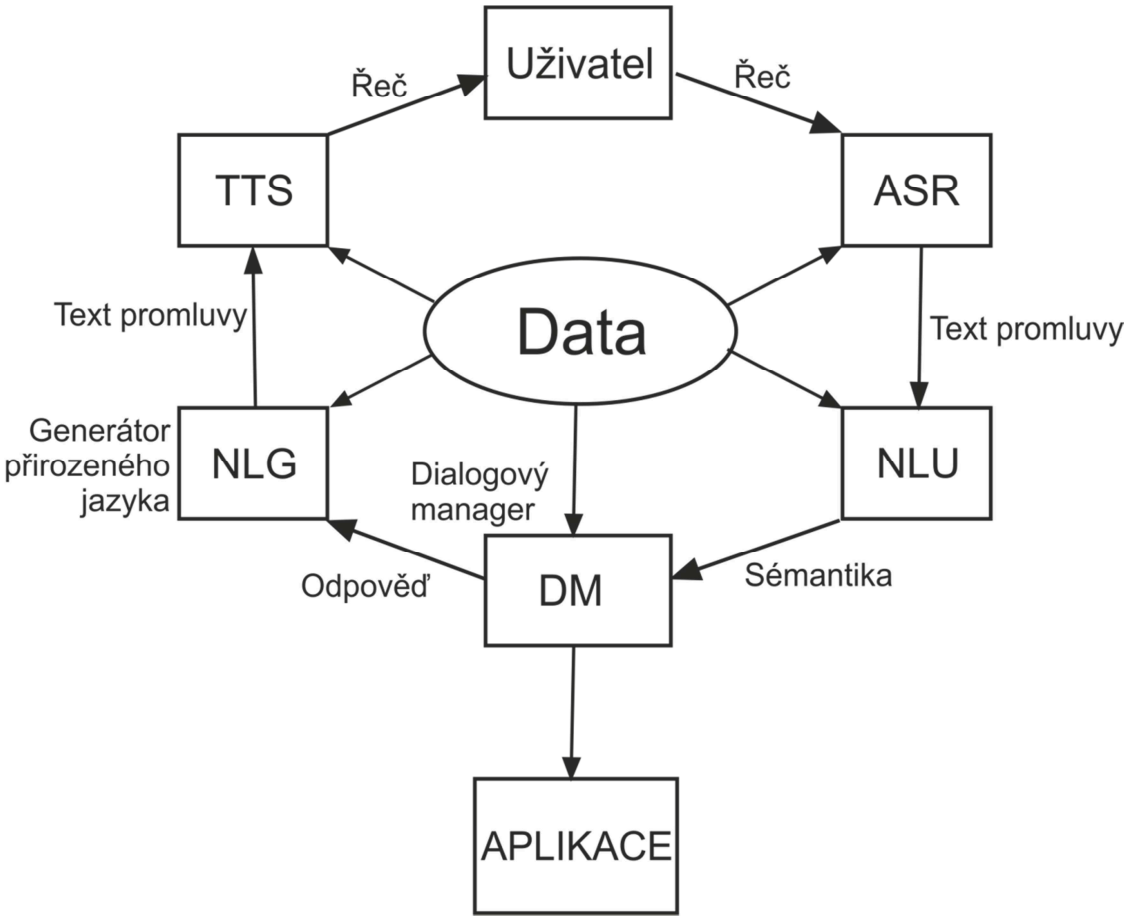
2.3.3 Vstupní dekodér a výstupní generátor

Vstupní dekodér rozeznává hlasový vstup za pomocí technologií zpracování lidské řeči, detekcí gest nebo rozpozná ručně psaný text. Poslední dvě jmenované technologie spadají do kategorie zpracování obrazu.

Výstupem může být čistý text nebo syntetizovaný hlas pomocí Text To Speech (TTS).

2.4 Dialogové systémy v praxi

Dialogové systémy jsou aplikovány především jako objednávkové systémy, kde si uživatel pomocí hlasového dialogu objednává různé služby např. letenky, jídla apod. Systém s uživatelem ihned po zavolání naváže dialog a za pomocí dotazů a různých informací od něj získá potřebná data pro zpracování a vytvoření objednávky, tu potom pošle dále ke zpracování.



Obrázek 2.3: Blokové schéma dialogového systému

3 VoiceXML

Je značkovací jazyk určený k popisu interaktivních dialogů mezi člověkem a počítačem, je vyvinutý na bázi XML, který vyvíjí konsorcium W3C. VoiceXML je určený k popisu scénářů pro automatizovaný dialog mezi člověkem a počítačem, je navržen pro hlasové dialogy s rozpoznáváním mluveného vstupu od uživatele a syntetizací výstupu. Umožňuje také rozpoznávat DTMF volby a vytvářet telefonní spojení. Lze pomocí něj popsat informace, které je potřeba získat, a komunikační scénáře, jak tyto informace získat. Obsahuje výstup ve formě zvukových souborů, které mohou nahradit syntetizovaný výstup, ten je tvořen pomocí modulů Text-To-Speech. [5]

3.1 Evoluce vývoje

Vývoj jazyka VoiceXML začal v roce 1995 ve společnosti AT&T Bell výzkumným projektem PhoneWeb, který vedli pánové Dave Ladd, Chris Ramming, Ken Rehor a Curt Tuckey. Když se společnost Lucent oddělila od AT&T, rozdělili se i zakladatelé projektu, Ramming zůstal v AT&T, Rehor odešel do společnosti Lucent a pánové Ladd a Tuckey přešli ke společnosti Motorola. Projekt pokračoval dále, ovšem každá společnost se ubírala jiným směrem, což vedlo také k vývoji odlišných jazyků. [6]

V roce 1999 zmíněné společnosti založily organizaci VoiceXML Forum, ke kterému se později přidala i firma IBM. Zhruba v polovině téhož roku byla vydána předverze jazyka označena jako 0.9, finální verze pak vyšla v březnu roku 2000, známá jako VoiceXML 1.0. VoiceXML Forum posléze předalo správu standardu do rukou konsorcia W3C. [7]

V roce 2003 vydalo konsorcium novou verzi jazyka a to VoiceXML 2.0, kde opravilo pár chyb z předchozí verze, ale hlavně byly přidány nové standardy a gramatiky, a také podpora rozeznání řeči a Text-To-Speech značkování. [8]

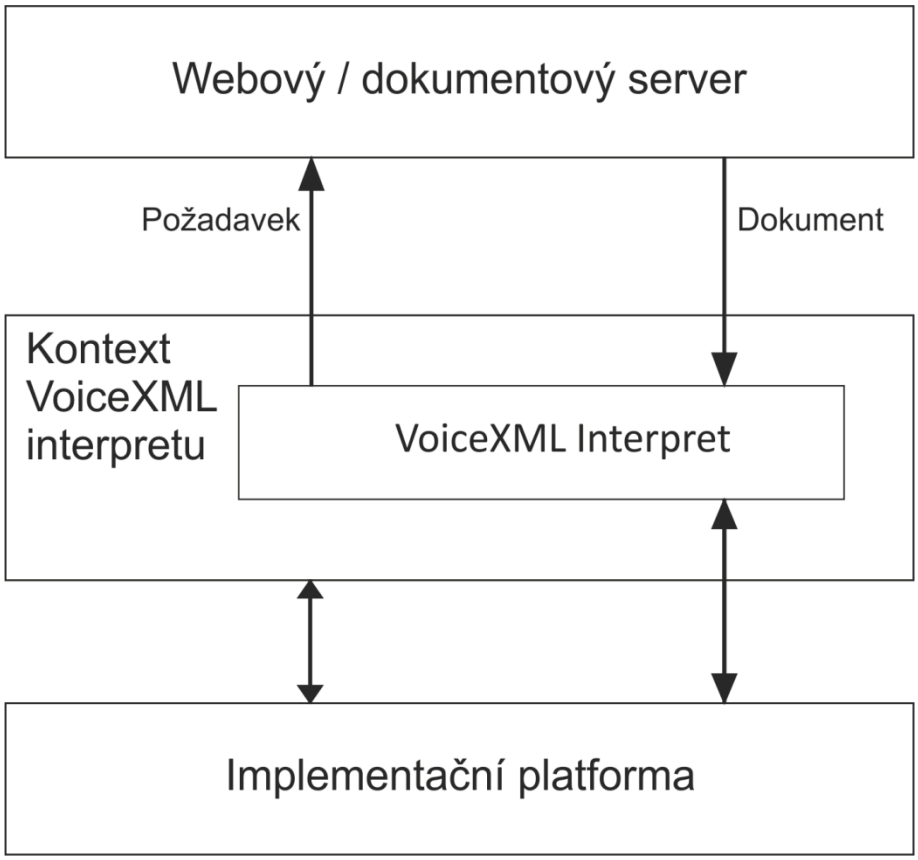
Verze 2.0 byla velice úspěšná, a podpořila vývoj mnoha hlasových aplikací, kterých bylo stále více. V červnu 2007 vydalo W3C verzi VoiceXML 2.1, ve které přibylo mnoho inovativních funkcí. [9]

Zatím poslední známá verze jazyka VoiceXML je verze 3.0, která byla vypuštěna do světa v prosinci roku 2010. Přináší mimo jiné nové zpracovávání médií – videa, hlasu, navigace a jiné. [5]

3.2 Architektura VoiceXML

Základní architektura VoiceXML je rozdělena do několika komponent. První je webový nebo dokumentový server, ten zpracovává požadavky od VoiceXML interpreta přes Kontext VoiceXML interpretu a vytváří odpovědi buď přímo jako VXML stránky, nebo jako server skripty, např. Perl, ASP, PHP nebo JSP, které jsou použity pro vytvoření VoiceXML odpovědi. Druhou komponentou je

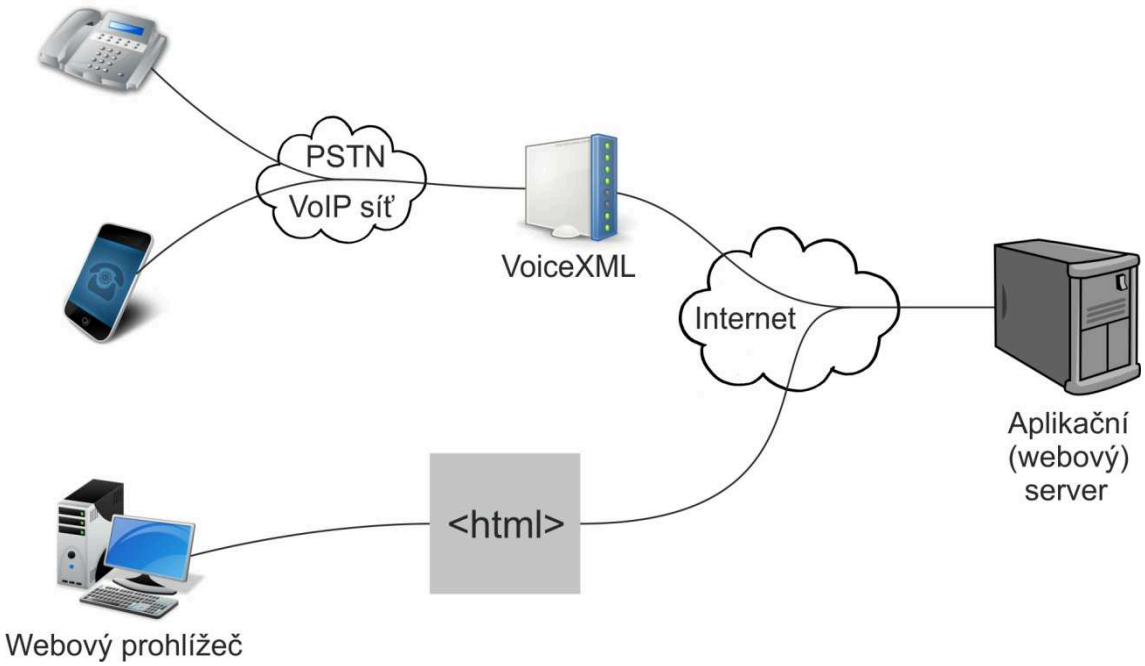
Kontext VoiceXML interpretu obsahující VoiceXML interpret, který je zodpovědný za interpretaci VXML kódu. Poslední částí je implementační platforma, která spravuje připojení k veřejné telefonní síti (PSTN), provádí rozpoznávání hlasu, přehrávání zvukových souborů a dalších. [10]



Obrázek 3.1: Architektura VoiceXML

3.3 Architektura aplikace

Architektura nasazení aplikace se skládá z webového nebo aplikačního serveru, serveru VoiceXML a klientských zařízení. Klienti VoiceXML telefonního serveru jsou připojeni buď přes PSTN nebo přes VoIP síť, servery a klienti s počítači jsou připojeni přes TCP/IP síť a používají k přenosu protokol HTTP.



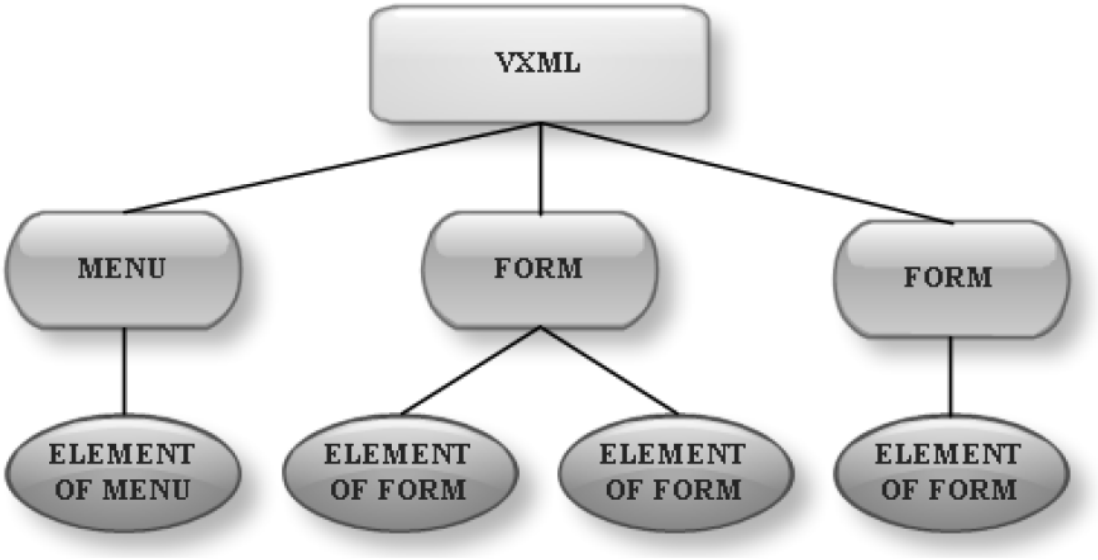
Obrázek 3.2: Schéma architektury aplikace VoiceXML

Aplikační server řídí základní aplikační logiku, má přístup k databázím, případně k transakčnímu serveru. Server VoiceXML je v pozici klienta Aplikačního serveru, obsahuje VoiceXML interpret, rozpoznávač a syntetizátor řeči. Uživatel, který přistupuje k informacím pomocí svého osobního počítače, získává data klasickou cestou prostřednictvím webových stránek ve formátu HTML pomocí protokolu HTTP. Zadá webovou adresu a klikáním myší na odkazy si vybere informaci, kterou si přeje zobrazit. Tato data se mu zobrazí na monitoru počítače. Uživatel, který bude přistupovat k informacím pomocí telefonu, zadá telefonní číslo, se kterým je služba spojena. Pomocí hlasových příkazů, nebo DTMF volby si zvolí požadované informace. VoiceXML interpret za pomoci gramatik rozpozná uživatelovo přání, a zašle požadavek na server. Ten požadavek zpracuje a zašle zpět VoiceXML dokument, který bude uživateli interpretován buď ve formě syntetizované řeči, nebo zvukovou nahrávkou. [11]

3.4 Struktura dokumentu

VoiceXML, jak již bylo zmíněno, vychází z jazyka XML, což je metajazyk vyvinutý konsorciem W3C a je standardem pro všechny značkovací jazyky. Aplikace VoiceXML je kolekce VoiceXML souborů, které jsou vzájemně propojeny a odkazují na sebe, je tak velmi podobný jazyku HTML. Na prohlížení webových stránek je potřeba použití webového prohlížeče, stejně tak se na čtení VoiceXML dokumentů používá interpret. Vývoj i práce s jazykem je zlehčena, protože se nemusí brát v úvahu programové vybavení ani operační systém, na kterém by měla být aplikace

provozována. Je to dáno volně přístupným, jednoduchým a standardizovaným formátem jazyka, jehož obsah je textový, nezávislý na kódování a je rozdělen značkami.



Obrázek 3.3: Struktura dokumentu VoiceXML [12]

VXML dokument se skládá z párových značek, ty mohou obsahovat atributy, které mají význam doplňující informace. Z obrázku 3.3 je vidět, že struktura dokumentu je stromová, to vede k jednodušší tvorbě a zpracovávání dokumentu. VoiceXML aplikace obsahuje jeden nebo více dokumentů, které mají společný kořenový dokument. Vždy, když uživatel v komunikaci aktivně využívá nějaký dokument v aplikaci, je aktivní i kořenový dokument. Také při přechodu uživatelů mezi jednotlivými dokumenty zůstává kořenový dokument stále aktivní. Zavírá se v okamžiku ukončení celé aplikace, takže pokud chceme někam uložit proměnné, které chceme mít po celou dobu načtené, kořenový dokument je tím vhodným místem. Tyto proměnné jsou pak také přístupné i ostatním dokumentům aplikace.

Každý VXML dokument je též XML dokumentem, a proto tedy musí začínat standardní XML hlavičkou. Kořenovým elementem dokumentu je element vxml, uvnitř je dokument rozčleněn do formulářů. Každý dokument by měl obsahovat tyto elementy:

Menu je nabídka, kde může uživatel vybírat ze seznamu uvedené parametry

Form je jeden z klíčových elementů dokumentu, pomocí něj probíhá výměna informací mezi uživatelem a aplikačním serverem

Block se používá ke stanovení akcí (výpis informací, odeslání dat na server, odkaz na jiný dokument, …), které jsou provedeny v pořadí daném dokumentem

Promt slouží k přehrání předem vytvořené zvukové stopy, nebo po vyplnění textem, je tento text syntetizován pomocí TTS modulu

Goto slouží jako přepínač mezi jednotlivými formuláři, v závislosti na jejich identifikátoru

Field slouží k získávání uživatelských vstupů

Nomatch vypíše zprávu, pokud nebyla rozeznána jeho odpověď

Submit odešle informace na server

Grammar odkazuje na gramatiku

To byl výčet několika základních elementů, nyní krátký příklad:

```
<?xml version="1.0" encoding="UTF-8"?>
  <vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">
    <form>
      <block>
        <prompt>Dobrý den, dovolali jste se na helpdesk
   firmy XY</prompt>
      </block>
      <field name="menu">
        <grammar type="application/srgs+xml" src="gram.grxml"/>
        <prompt>Přejete si přepojit na operátorku, servisního
   technika nebo na záznamník?</prompt>
        <nomatch>Zadejte operátorka, servisní technik
   nebo záznamník.</nomatch>
      </field>
      <block>
      <prompt>Zvolil jste <value expr="menu">, nyní budete
 přepojeni.</prompt>
      <submit next="server.jsp" namelist="menu"/>
      </block>
    </from>
  </vxml>
```

Dokument nejdříve vypíše uvítání a potom se zeptá kam má zákazníka přepojit. Pokud se systému nepodaří rozeznat uživatelovu odpověď, tak mu poskytne nápovědu. Pokud odpověď rozezná, oznámí volbu a další krok, pak odešle volbu na server. Pro rozeznání odpovědí slouží gramatika *gram.grxm*, která je umístěna někde v externím souboru a je na ni odkazováno pomocí elementu <*grammar*>.

VoiceXML také zčásti zabezpečuje ovládání směrování hovoru. Používá pro to vyhodnocování podmínek <*if*>, <*elseif*> a <*else*>. Jazyk obsahuje také nástroje na zpracování některých problémových stavů. Případ, když uživatel nezadá žádnou volbou se ošetří elementem <*noimput*>, pokud uživatel zažádá o pomoc, vykoná se akce v elementu <*help*>. V neposlední řadě také VoiceXML nabízí možnosti správy výjimek, a to elementy <*throw*> a <*catch*>. [5] [12] [13]

3.5 Související technologie

S problematikou hlasových aplikací a dialogových systémů souvisí vícero technologií a standardů, z kterých je většina pod správou konsorcia W3C. Některé z nich zabezpečují ovládání zpracování souborů a hovorů, jiné zase souvisí se zpracováním řeči, jejím rozpoznáním a syntézou.

3.5.1 Technologie ovládání zpracování

Tyto technologie jsou určené pro synchronizaci jednotlivých prvků v dané aplikaci nebo dialogovém systému.

CCXML (Call Control XML) je značkovací jazyk vyvinutý konsorciem W3C a slouží pro kontrolu telefonních hovorů. Zabezpečuje správu, ovládání a převody telefonátů prostřednictvím síťové vrstvy. Jeho úlohou je informovat interpreta o tom, jak má zpracovávat informace o ovládání hovoru z hlasového kanálu. CCXML je možné použít i mimo aplikaci VoiceXML, používá se také v systémech interaktivní hlasové odezvy IVR, kde zabezpečuje ovládání přepojování hovorů. [14]

SCXML (State Chart XML) je značkovací jazyk pro popis stavů systémů kombinující CCXML a Harelovy automaty. V kontextu aplikací VoiceXML je nástupcem CCXML a používá se k navigaci mezi různými dokumenty a logickými částmi aplikací. [15]

Jazyk **ECMAScript** je ovlivněný jazyky Java, Perl, Python a Hypertalk a v dialogových systémech se využívá hlavně k vyhodnocování výrazů a gramatik. Na propojení s dokumenty XML se používá rozšíření jazyka E4X (ECMAScript for XML). Běžně se při zpracování XML dokumentu používá převedení XML do modelu DOM (Dokument Object Model). Rozšíření E4X nepoužívá toto převedení, ale primitivní přístup urychlující tvorbu programu a práci s dokumenty XML. Jazyk ECMAScript je implementovaný například v jádře Gecko, což je základ pro Mozillu Firefox. [16]

3.5.2 Technologie zpracování řeči

Mezi tyto technologie patří několik jazyků popisujících způsob zpracování a formu dat. Jsou používané při samotném zpracování řeči, při jejím rozpoznání a syntéze.

SSML (Speech Synthesis Markup Language) Popisuje způsob a parametry syntézy řeči, např. druh syntetizátoru, rychlost, tón a klesání hlasu. [17]

SRGS (Speech Recognition Grammar Specification) udává rozeznávání řeči, jaké řetězce jsou přípustné pro daný jazyk. Na popis různých řetězců používá gramatiky. [18]

SISR (Semantic Interpretation for Speech Recognition) extrahuje sémantický význam z údajů získaných pomocí rozeznávání řeči. Úzce souvisí s SRGS. V praxi se využívá v kombinaci s programovacím jazykem ECMAScript. [19]

PLS (Pronunciation Lexicon Specification) je specifikace konsorcia W3C, poskytuje prostředky k popisu výslovnosti v syntetizačním procesu. Vytváří spojení mezi zapsanými slovy a jeho výslovností použitím fonetické abecedy. Používá se v případech, kdy výstupní text obsahuje znaky nebo slova, která standardně může TTS zařízení špatně syntetizovat nebo pokud se výslovnost slov liší podle jeho pozice ve větě. [20]

Syntéza řeči je umělé vytváření zvuků lidské řeči, využívá jazyk SSML. Syntézu vykonává syntetizátor a může být implementován buď v programovém, nebo v technickém vybavení.

Převod textu na řeč vykonávají TTS moduly a jejich součástí jsou syntetizátory řeči.

Rozpoznávání řeči je proces, při kterém se zpracováváním zvukového vstupu získává řada slov, používají se k tomu technologie SRGS, SISR a PLS.

4 Vybrané komponenty VoiceXML aplikace

VoiceXML aplikace se skládá z mnoha částí, jednou z nich je pobočková ústředna Asterisk, ve které je nainstalován správce řečových syntéz Festival. Další důležitou součástí je interpret. V dnešní době existuje několik desítek interpretů pro VoiceXML, většina z nich je proprietární bez možnosti vyzkoušení. Vzhledem k velkému množství interpretů a zaměření této práce představím jen dva, a to VXIasterisk a VoiceGlue.

4.1 Asterisk

Asterisk je open source telefonní ústředna vyvíjená společností Digium a je vydávána pod licencí GNU GPL. Asterisk sám o sobě nemá v základním provedení podporu VoiceXML implementovánu. Asterisk je díky své flexibilitě a modularitě použitelný v mnoha různých úlohách. Jako pobočková telefonní ústředna PBX (Private Brach eXchange) vytváří spojení IP telefonie s veřejnou telefonní sítí. Umožňuje spojování hovorů a konferenčních hovorů. Asterisk ve funkci brány může navzájem propojit různé druhy telefonních sítí a převádí data mezi jednotlivými protokoly. Dále je možné jej použít jako server interaktivní hlasové odezvy (IVR) zabezpečující konferenční mosty a automatickou obsluhu volajících. [21]

Asterisk je navržený pro platformy Linux, OpenBSD, FreeBSD a Mac OS. Integrace s jinými Unixovými systémy taktéž není problematická. V současné době je nejnovější dostupná verze 11.0.1 vydaná 25.10.2012. Veškerá dokumentace k Asterisku je dostupná online zdarma ke stažení, stejně jako samotná aplikace.

Asterisk má přímou podporu pro rozšíření zařízení pro telefonii, kromě zařízení vybraných výrobců podporuje všechny zvukové karty, které podporují ALSA a OSS. Dále podporuje standard isdn4linux, a proto je možné pomocí něj zprovoznit koncové zařízení ISDN, které tento standard podporují.

Asterisk podporuje tyto porotokoly:

- IAX (Inter-Asterisk Exchange)
- H.323
- SIP (Session Initiation Protocol)
- MGCP (Media Gateway Control Protocol)
- SCCP (Skinny Call Protocol)

Kodeky, které je Asterisk schopen používat jsou uvedeny v tab. 4.1

Tabulka 4.1: Používané kodeky

Kodek	Šířka pásma
G.711	64 kbps
G.726	16, 24 nebo 32 kpbs
G.723.1	5,3 nebo 6,3 kbps
G.729A	8 kbps
GSM	13 kbps
iLBC	13,3 kbps nebo 15,2 kbps
Speex	Proměnná

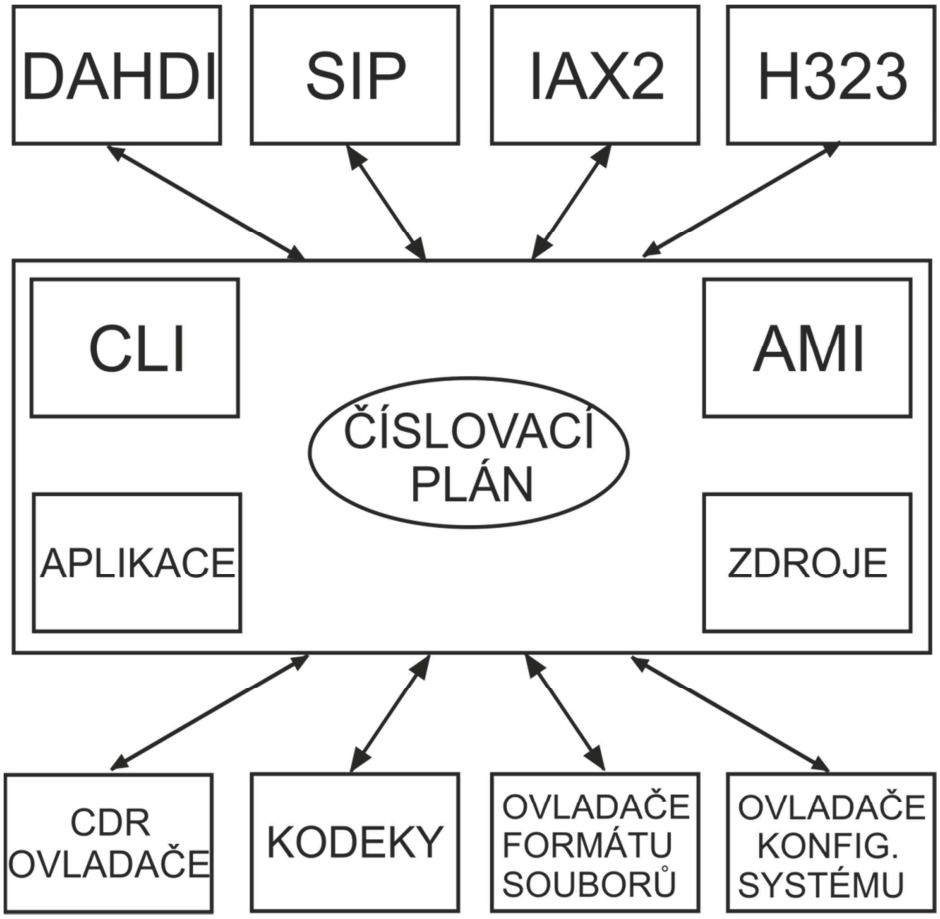
4.1.1 Základní části Asterisku

Asterisk je tvořen mnoha moduly, konfiguračními a zvukovými soubory. Základní nastavení Asterisku je uloženo v souboru **asterisk.conf**. Ten je rozdělen do několika částí:

- **Directories** – umístění používaných adresářů
- **Options** – výchozí nastavení příkazové řádky
- **Files** – nastavení pro vzdálenou konzoli
- **Compat** – nastavení zpětné kompatibility Asterisku

V modulární architektuře jsou jádro a moduly úplně abstrahovány od použitých technologií. Toto umožňuje jednodušší použití Asterisku s různým technickým a programovým vybavením. Pro komunikaci s moduly disponuje jádru čtyřmi rozhraními, Kanálové API, Aplikační API, API pro práci se soubory a API pro překlad kodeků. Vnitřní architektura Asterisku je znázorněna na obrázku 4.1

Kanálové API slouží ke zpracování připojení, kterým se připojují volající ze sítí ISDN, VoIP a jiných. Přidává další moduly pro nízkoúrovňové zpracování daného připojení a použitého kodeku. Aplikační API umožňuje připojení dalších aplikací jako aplikace pro podporu hlasových zpráv nebo VoiceXML. Zpracování souborů a přístup k nim má na starosti API pro práci se soubory. API překladu kodeků umožňuje konverzi mezi zvukem v různých kódováních. Pro každé API je v jádře interní modul, přepojení hovoru za podpory kanálového API zabezpečí ústředna PBX. Přístup k souborům za podpory API pro práci se soubory má na starost plánovač a správce vstupů a výstupů. Nahrávání aplikačních modulů a spouštění aplikací spravuje spouštěč aplikací. Největší využití šířky pásma a nejvyšší kvalitu zvuku zabezpečí překladač kodeků. Ten nahrává moduly pro jednotlivé kodeky a vykonává převod mezi nimi. [21] [22]

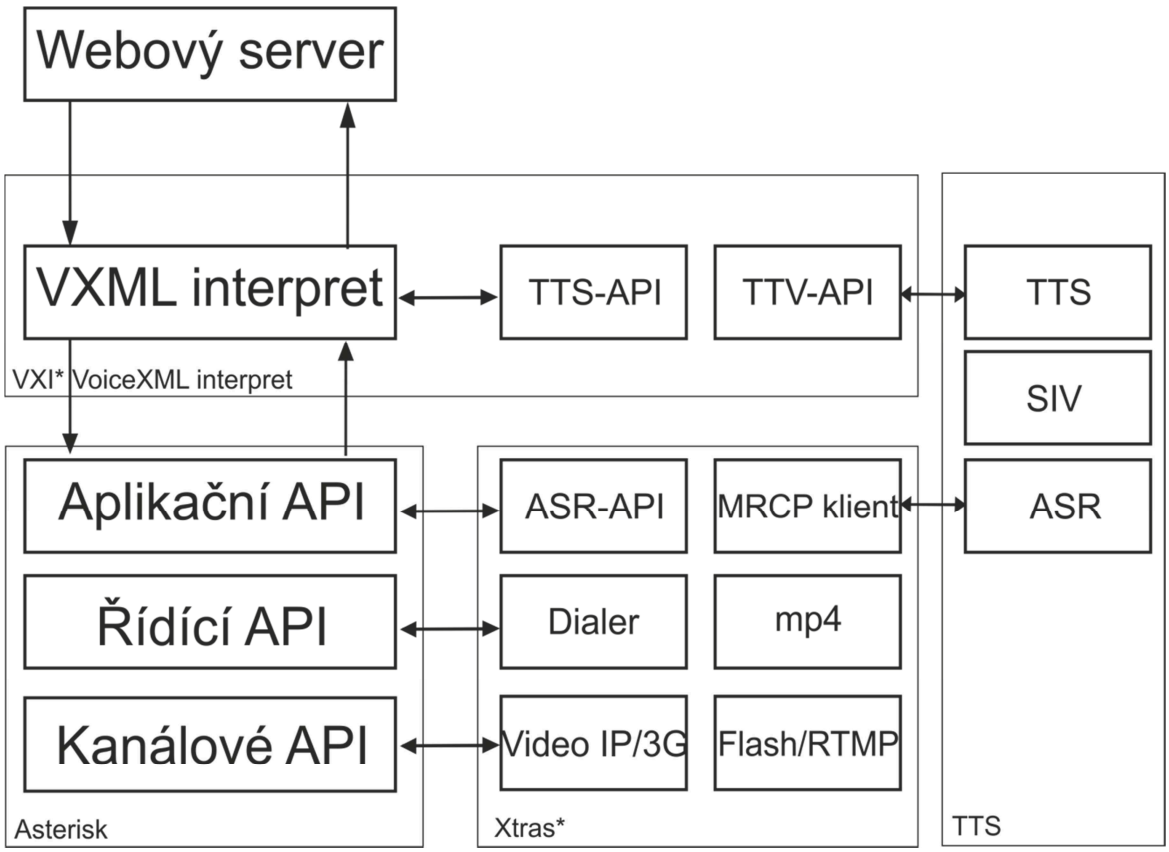


Obrázek 4.1: Architektura Asterisku [23]

4.2 VXI*

VXI* je oficiální nativní integrace Asterisku pro podporu VoiceXML. Současná verze 7.1 vydaná v listopadu 2012 společností I6NET podporuje jádra Asterisku 1.4, 1.6 a 1.8. VXI* je distribuovaný jako komerční aplikace s potřebou zakoupení licence pro použití více jak jednoho portu. Jeden port je pro zkušební aplikaci neomezeně zdarma, každý další pak stojí 100€. Obsahuje podporu webových protokolů VoiceXML 2.0+, MRCPv1 a v2, SRGS, SSML, RTSP, RTMP HTTP a HTTPS. Dále nechybí rozpoznávač řeči ani syntetizátor řeči, podpora zpracování různých medií, zvuk, video, videokonference. Podporované jsou linuxové operační systémy založené na distribuci Debian s jádrem 2.6 a vyšším. Při psaní této kapitoly bylo čerpáno z následujícího pramenu [24]

Společnost I6NET se zabývá vývojem interaktivních komunikačních aplikací, vytváří VoiceXML komponenty, které umožňují vznik pokročilých hlasových a video služeb. Na obrázku 4.2 je znázorněno blokové schéma VoiceXML aplikace postavené na VXI*. Je zapotřebí několika různých komponent, ze kterých dva přímo vyvíjí společnost I6NET, jsou to VXI* interpret VoiceXML a Xtras* Software extensions.



*Obrázek 4.2: Architektura VoiceXML aplikace s využitím VXI**

Aplikace VXI* obsahuje několik důležitých funkcí:

Rozpoznávání řeči – VXI* obsahuje několik modulů rozpoznávání řeči, ty jsou schopny se k Asterisku připojit pomocí nativního API, nebo pomocí MRCPv1 či MRCPv2 skrze uniMRCP.

Text-to-Speech – aplikace podporuje mnoho nástrojů převodu textu na řeč, využívají cache paměť pro zlepšení výkonu převodu. TTS může být připojen i vzdáleně, podporovány jsou servery:

- ScanSoft / Nuance
- Acapela
- Loquendo
- Ivona
- Verbio
- Orange Baratinoo
- Cepstral
- Voice Interaction (Audimus)
- Flite / eSpeak / MBROLA
- Google Text-to-Speech

Text-to-Video (TTV) – rozšiřující video funkce umožňující generovat video z textu, např. generování grafického rozbalovacího menu na displej telefonu

Ověření a identifikace volajícího – VXI* dokáže pracovat s biometrickým aplikačním rozhraním mnoha telefonů, obsahuje VoiceXML rozhraní, které umožňuje identifikaci a ověření volajícího

Modul Xtras* je volitelný rozšiřující modul pro aplikaci VXI* a Asterisk. Toto rozšíření přináší nové funkce, jako jsou:

- Outbound Dialer/API
- Call Recorder MP4
- MRCP Client
- Flash/RTMP Server Channel
- Video IP/3G and Tools

Pro praktickou realizaci své úlohy jsem použití aplikace VXI* vyloučil z několika následujících důvodů:

- Aplikace není open source, není pod GNU GPL licencí
- Aplikaci nemůže využít současně více klientů
- Obtížné nasazení systému řečových syntéz Festival

4.3 VoiceGlue

VoiceGlue je další rozšíření pro Asterisk umožňující zpracovávání aplikací VoiceXML. Je založen na open source řešeních a je vydáván pod licencí GNU GPL, využívá OpenVXI interpret. Je možné jej stáhnout ve formě balíku tar.gz ze stránek autorů. V distribuci je zahrnut syntetizátor Flite, VXML 2.0 interpret s některými funkcemi VXML 2.1, SRGS DTMF gramatiky a další. Dokumentace je přístupná taktéž online na webových stránkách.

Projekt VoiceGlue vznikl za účelem získat levnější řešení VoiceXML aplikací. Hlavím autorem a vývojářem je Doug Campbell ze společnosti Ampersound. První verze projektu podporovala jen DTMF vstup a výstup byl jen ve formě namluveného zvukového souboru. Postupem času začaly přibývat podpory modulů převodu textu na řeč a rozpoznávání řeči. Zaměstnanci Ampersound se začali zajímat o integraci aplikace s PBX Asterisk a OpenVXI. Tohoto cíle dosáhli za pomocí open source aplikací, a proto se rozhodli pracovat na verzi pro všeobecné použití. [25]

4.3.1 Vlastnosti a funkce VoiceGlue

V současné verzi 0.14 je použit VoiceXML interpret OpenVXI 3.4 a jsou podporovány tyto funkce:

- Interpretace VXML dokumentů 2.0 s některými funkcemi VXML 2.1
- Syntéza řeči pomocí modulu Flite
- Přehrávání všech zvukových souborů podporovaných Asteriskem
- Detekce DTMF
- SRGS DTMF gramatiky
- Nahrávání ve formátu ulaw
- Sdílená mezi-paměť stažených zvukových souborů
- Sdílená mezi-paměť volání TTS modulu
- Sdílená mezi-paměť SRGS DTMG gramatik
- Rozdělitelné komponenty zpracovávání

Plánované funkce:

- Rozpoznávání řeči pomocí nástroje LumenVox
- Podpora všech funkcí VXML 2.1
- Doplnění dalších zvukových formátů pro přehrávání a nahrávání
- Integrace kvalitnějšího TTS modulu

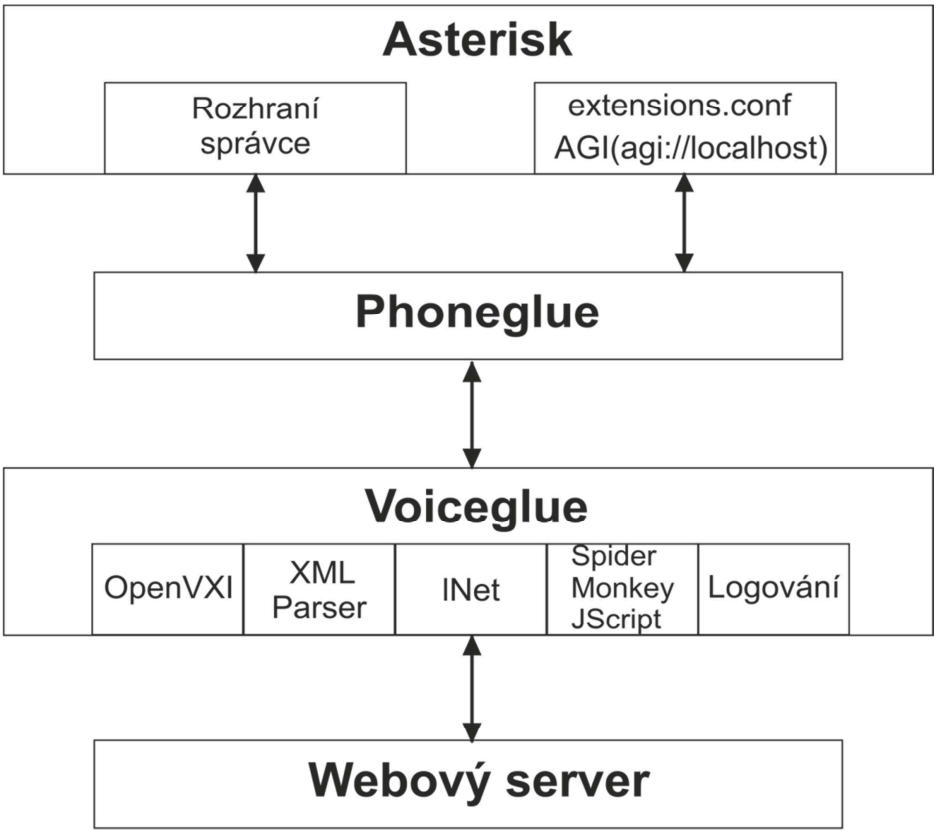
4.3.2 OpenVXI

Je knihovna potřebná k interpretaci značkovacího dialogového jazyka VoiceXML, která přísně dodržuje jeho specifikaci. Poskytuje aplikační rozhraní pro rozpoznávání řeči, syntézu řeči a další telefonické funkce. OpenVXI je nezávislá na platformě a poskytuje pevný základ k integraci ostatních komponent aplikace. Tuto knihovnu vyvinula původně společnost SpeechWorks International, ale od poloviny roku 2004 jí spravuje společnost Vocalocity a je licencována pod GNU GPL.

4.3.3 Architektura VoiceGlue

Výhodou je podpora distribuovaného zpracování, VoiceGlue nemusí běžet na tom samém serveru jako Asterisk. Je určený pro běh na Asterisku, a neměl by být problém jej zprovoznit na jakémkoli Unixovém systému.

VoiceGlue je možné logicky rozdělit na tři hlavní části – rozhraní telefonie, část zpracovávající VoiceXML dokumenty a na síťovou část, viz obrázek 4.3



Obrázek 4.3: Architektura VoiceGlue

Rozhraní pro telefonii zahrnuje několik částí, rozpoznávač řeči, syntetizátor řeči Flite a PhoneGlue, který pomocí zpráv SATC komunikuje s VoiceGlue a pomocí rozhraní FastAGI nad protokolem TCP/IP komunikuje s Asteriskem. Část zpracovávající VoiceXML dokumenty se skládá z více modulů. Před zpracováním dokumentů VoiceXML interpretem OpenVXI jsou tyto nejdříve rozparsovány syntaktickým analyzátorem Xerces. Případné skripty jazyka ECMAScript jsou spravovány modulem Spider-Monkey, poslední částí je logovací modul pro vytváření záznamů. Síťová část komunikuje s webovým server obsahující VoiceXML dokumenty, které přenáší protokolem HTTP. [26]

4.3.4 **Obsah distribuce VoiceGlue**

Po stáhnutí a rozbalení archívu obsahující VoiceGlue distribuci se v hlavním adresáři nachází několik podadresářů. Nejdůležitější z nich je adresář *doc*, který obsahuje skripty určené k nainstalování nebo odinstalování VoiceGlue, jsou ovšem určené jen pro některé distribuce operačního systému Linux, v případě jiné distribuce je potřeba provést kompilaci a instalaci ručně. Dále se zde nachází soubor *README* se stručnými informacemi a odkazem na webový návod k instalaci. Poslední položkou je adresář *examples*, který obsahuje ukázkové VoiceXML dokumenty.

Ostatní podadresáře v kořenovém adresáři jsou jednotlivé součásti VoiceGlue, které obsahují knihovny a programy, viz tabulka 4.1

Tabulka 4.2: Součásti VoiceGlue

Cam-Scom	knihovna pro IPC
dynlog	program sloužící k záznamu událostí
libvglue	knihovna VoiceGlue
openvxi-3.4+vglue	upravená knihovna OpenVXI 3.4
phoneglue	program poskytující vysoko-úrovňové rozhraní Asterisku
Satc	knihovna pro komunikaci s phoneglue
SRGSDTMF	knihovna pro zpracování gramatik
voiceglue	program poskytující VoiceGlue rozhraní
Voiceglue-Conf	knihovna pro konfiguraci VoiceGlue
Vxglue	knohovna pro komunikaci s OpenVXI

4.4 Správce řečových syntéz Festival

Jedná se o syntetizační program převodu textu na řeč, tzv. text-to-speech, který byl vyvinut na Univerzitě v Edinburghu. Je psaný v jazyce C++ a obsahuje knihovnu Edinburgh Speech Tools. Festival je vícejazyčný, nejrozšířenějším jazykem je angličtina ovšem pomocí projektu FestVox lze budovat nové hlasy téměř v jakémkoli jazyce.

Česká difonová databáze pro systém syntézy řeči Festival obsahuje jeden český hlas s názvem voice-czech-ph. Na jeho vývoji se podíleli společnosti Brailcom, o.p.s, která se zabývá zprostředkováváním důležitých informací nevidomým a slabozrakým. Dále Nadační fond Českého rozhlasu, Seznam.cz, a.s. a Evropská komise – program Leonardo da Vinci. Odborné znalosti a konzultace poskytl Fonetický ústav Filozofické fakulty Karlovy Univerzity v Praze pod vedením Prof. PhDr. Zdeny Palkové, CSc. Další tři české hlasy s názvy voice-czech-krb, voice-czech-dita a voice-czech-machac vytvořil PhDr. Pavel Machač, Ph.D. a Mgr. Radek Skarnitzl, Ph.D. z Ústavu fonetiky Fakulty Filozofie Univerzity Karlovy v Praze za technické podpory společnosti Brailcom, o.p.s. Tyto hlasy byly vytvořeny v rámci projektů „Olomoucká šance pro nevidomé žáky“ a „Centrum podpory pro zrakově postižené žáky“, které jsou spolufinancovány z Evropského sociálního fondu a ze státního rozpočtu České republiky.

5 Instalace aplikace

V této části bude popsána instalace a zprovoznění požadovaných programů pro tvorbu VoiceXML aplikace.

5.1 Příprava operačního systému

Hlavní operační platformou pobočkové ústředny Asterisk je Linux. V rámci kompatibility VoiceGlue verze 0.14 s ostatními komponentami jsem zvolil operační systém Linux Ubuntu, dlouhodobě podporovanou verzi 10.04 a Asterisk ve verzi 1.8. Na nově nainstalovaný Ubuntu server bylo ještě nutné nainstalovat webový server Apache s podporou PHP5 následujícím příkazem:

```
# apt-get install apache2 libapache2-mod-php5 php5
```

5.2 Instalace a konfigurace Asterisku

V prvním kroku je potřeba stáhnout aktuální verzi Asterisku. Vytvořil jsem si složku *download* v adresáři */opt,* do které jsem balík stáhnul a následně rozbalil.

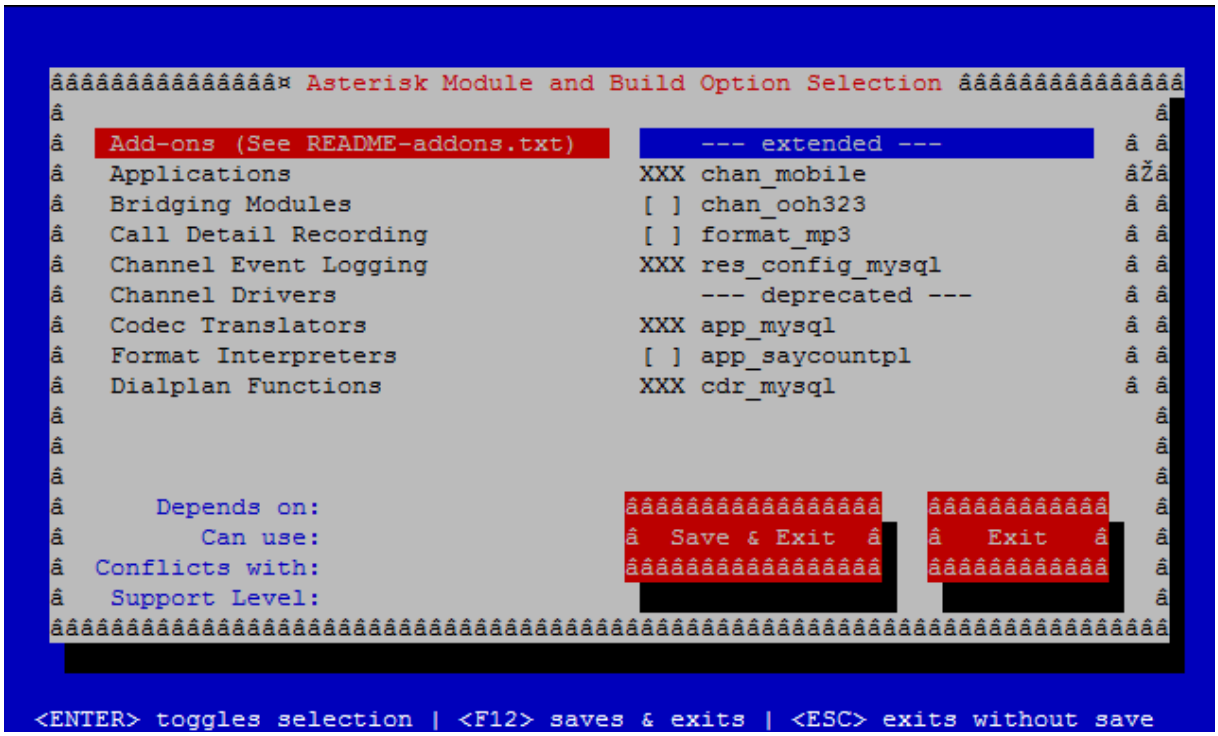
```
# mkdir /opt/download && cd /opt/download
# wget http://downloads.asterisk.org/pub/telephony/asterisk/ \
asterisk-1.8- current.tar.gz
# tar –xvf asterisk-1.8-current.tar.gz
```

Zkompilování Asterisku jsem provedl v rozbaleném adresáři, nejdříve příkazem *./configure*, kdy systém upozorňuje na případné chybějící balíčky, které je potřeba pro úspěšné dokončení nainstalovat. Pokud je vše v pořádku, tak provedení příkazu trvá přibližně minutu a jeho správné ukončení signalizuje textové vyobrazení loga Digium. Poté je zapotřebí zvolit jaké moduly se mají při instalaci Asterisku nainstalovat příkazem *make menuselect*. Moduly, ovladače a jiné funkce lze vybrat pomocí grafického menu, viz obrázek 5.1 Na pravé straně se nachází výpis kategorií a na levé straně jsou vypsány jednotlivé moduly patřící do vybrané kategorie.

Kompilaci jsem spustil příkazem `make` a následně příkazem *make install* jsem jej nainstaloval. Posledním instalačním krokem byl příkaz *make samples*, který vytvořil základní konfigurační soubory. Celý proces trval několik minut a po jejím ukončení bylo možné Asterisk spustit příkazem */etc/init.d/asterisk start.* Po spuštění je možné se připojit do konzole Asterisku příkazem *asterisk –cvvv*.

V poslední řadě jsem musel vytvořit uživatele asterisk ve stejnojmenné skupině, pokud bych takto neučinil, instalace Voiceglue, která je popsána níže, by se nezdařila.

```
# groupadd asterisk
# useradd asterisk -d /var/lib/asterisk/ -g asterisk asterisk
```



Obrázek 5.1: Grafické menu pro výběr modulů Asterisku

Konfigurační soubory Asterisku se nachází v adresáři `/etc/asterisk/`, pro základní otestování funkčnosti ústředny je zapotřebí vytvořit SIP účty v souboru `sip.conf` a následně vytvoření číslovacího plánu v souboru `extensions.conf`. Asterisk verze 1.8 umožňuje vytváření šablon, což velice zjednodušuje vytváření SIP účtů. Každý účet musí mít definovány povinné parametry, jakou jsou povolené kodeky nebo kontext do kterého patří. Pokud jsou tyto parametry stejné, lze vytvořit šablonu, kde se tyto parametry nadefinují a při vytváření účtu se na ně pak odkáže názvem šablony.

Ukázka definice uživatelů v sip.conf pomocí šablony:

```
[sablona](!)
context=vxml
type=friend
dtmfmode=rfc2833
disallow=all
allow=alaw
```

```
allow=ulaw
host=dynamic
[200](sablona)
username=200
secret=200
```

Takto jsem vytvořil SIP účty s názvy 200 – 205 a v číslovacím plánu jsem pod číslem 100 vytvořil aplikaci, která mi po zavolání přehraje slova „Hello world“.

```
[vxml]
exten => 100,1,Answer
exten => 100,n,Playback(hello-world)
exten => 100,n,Hangup
```

5.3 Instalace a konfigurace TTS Festival

Program Festival je v operačním systému Ubuntu 10.04 obsažen v oficiálním repozitáři. V základním balíčku programu Festival je obsažen jen jeden český hlas, jedná se o `voice-czech-ph`. Ovšem po přidání odkazu do repozitáře jsou dostupné i další tři české hlasy, jedná se o `voice-czech-krb`, `voice-czech-machac` a `voice-czech-dita`. Jejich instalaci provedeme následujícími příkazy:

```
# add-apt-repository ppa:dusek/voice-czech
# apt-get update
# apt-get install festival festival-czech festvox-czech-dita
```

Pro svou aplikaci jsem si zvolil hlas `voice-czech-dita`, který je dle mého názoru nejsrozumitelnější. Aby Festival prováděl syntézu českého textu na vybraný český hlas, musíme nakonfigurovat soubor `/usr/share/festival/festival.scm`. Na konec souboru přidáme definici češtiny a hlasu:

```
(set! server_access_list '("localhost\\.localdomain" "localhost"))
(require 'czech)
(require 'czech-unisyn)
(set! inhibit-initial-pauses t)
(set! voice_default 'voice_czech_dita)
(define (tts_textasterisk string mode)
(let ((wholeutt (utt.synth (eval (list 'Utterance 'Text string)))))
```

```
(utt.wave.resample wholeutt 8000)
(utt.wave.rescale wholeutt 5)
(utt.send.wave.client wholeutt)))
```

5.4 Instalace a konfigurace Voiceglue

Z oficiálních webových stránek projektu Voiceglue jsem si stáhl nejnovější balík, verze 0.14. Tato verze je kompatibilní s operačním systémem Ubuntu 10.04 a 11.04, pro nejnovější dlouhodobě podporovanou verzi 12.04 chystají vývojáři novou verzi Voiceglue, ovšem datum jejího vydání ještě není známo.

Před samotnou instalací VoiceGlue, je potřeba nainstalovat požadované balíčky, viz Tabulka 4.1 zobrazující názvy balíčků pro jednotlivé systémy.

Tabulka 5.1: Balíčky potřebné k nainstalování Voiceglue

Balíček	**Ubuntu 10.04 - 11.04**	**Fedora 12-13, CentOS 5.4 - 5.6**
gcc/g++	gcc g++	gcc-c++
libc++		libstdc++-devel
xerces-c	licxerces-c2-dev	xerces-c-devel
SpiderMonkey	xulrunner-dev	xulrunner-devel
flite	flite	Flite
sox	sox	Sox
libsox-fmt	libsox-fmt-base	libsox-fmt-all
curl	curl	Curl
opessl	libssl-dev	openssl-devel
XML::LibXML	libxml-libxml-perl	perl-XML-LibXML
BSD::Resource	libbsd-resource-perl	perl-BSD-Resource
Module::Build	libmodule-build-perl	perl-Module-Build
URI::Escape	liburi-perl	CentOS: perl-URI perl-URI.Escape- XS
Test::More		perl-Test-Base
pkg-config	pkg-config	pkgconfig

VoiceGlue lze nainstalovat dvěma způsoby. První a jednodušší způsob je spuštění příkazu *./doc/install-voiceglue* v rozbaleném archivu stažené aplikace. Tento způsob funguje jen v některých distribucích systému Linux, pro ostatní verze platí druhý, složitější způsob instalace. Pro účely této práce byl zvolen operační systém Ubuntu 10.04 LTS, ve kterém lze VoiceGlue instalovat prvním způsobem, ale pro úplnost uvedu i ten těžší způsob instalace.

Po stažení a rozbalení archívu jsem se přesunul do kořenového adresáře *voiceglue-voiceglue-acef2bb/*, ve kterém se nachází 11 podadresářů, včetně adresáře *doc*. Nyní je potřeba nainstalovat každou část aplikace zvlášť a ve správném pořadí.

Instalace knihoven – *Cam-Scom, libvglue, Satc, SRGSDTMF, Voiceglue-Conf* a *Vxglue*:

- Nastavení cesty do aktuálního adresáře knihovny
- Spuštění příkazů

```
# perl Build.PL
# ./Build install
```

Instalace programů *dynlog*, *phoneglue* a *voiceglue*:

- Nastavení cesty do aktuálního adresáře programu
- Spuštění příkazů (Debian)

```
# make -f Makefile.debian install
```

- Spuštění příkazů (Red Hat)

```
# make -f Makefile.redhat install
```

Při instalaci OpenVXI je potřeba upravit instalační skript, je nutné nastavit cestu k aplikaci *xulrunner*, našel jsem tedy v souboru *openvxi-3.4+vglue/build_openvxi* řádek:

```
$::ENV{"SPIDERMONKEYDIR"} = "/usr";
```

A nahradil jej řádkem

```
$::ENV{"SPIDERMONKEYDIR"} = "/usr/include/xulrunner-1.9.2.28/";
```

Následná instalace aplikace, příkazy:

```
# ./build_openvxi
# ./install_openvxi
```

Voiceglue bude přistupovat k Asterisku přes Phoneglue, proto je zapotřebí nastavit práva přístupu v souboru `/etc/asterisk/manager.conf`. Původní základní nastavení jsem ponechal a na konci konfiguračního souboru jsem vytvořil uživatele phoneglue.

```
[phoneglue]
secret=phoneglue
read = system,call,log,verbose,command,agent,user,originate
```

```
write = system,call,log,verbose,command,agent,user,originate
```

Dalším krokem po nastavení přístupu je vytvoření skriptu, který zajistí převod textu na řeč pomocí aplikace Festival. Využil jsem k tomu funkci s názvem text2wave, jehož argumenty jsou výstupní frekvence audio souboru, název výstupního .wav souboru a vstupní text nebo textový soubor. Voiceglue používá pro převod textu na řeč skript `/usr/bin/voiceglue_tts_gen`, který je potřeba upravit. Jedná se o skript psaný v jazyce PERL, výsledný kód vypadá následovně:

```
$text = $::ARGV[1];
$soubor = $::ARGV[2];
use Text::Iconv;
$converter = Text::Iconv->new("utf-8", "iso-8859-2");
$prevod = $converter->convert($text);
open (VTEXT, ">/tmp/".$text.".txt");
print VTEXT $prevod;
close (VTEXT);
system ("/usr/bin/text2wave", "-F", "8000", "-o", $soubor,
"/tmp/".$text.".txt");
```

XML Parser implementovaný ve Voiceglue projíždí VoiceXML dokument, pokud narazí na text, který se má přečíst, předá jej v argumentu výše zmíněnému skriptu. Vzhledem ke kódování VoiceXML dokumentu, který používá UTF-8 kódování, a kódování ISO-8859-2 používané v operačních systémech Linux, je třeba provést převod textu mezi těmito kódováními. Pokud by se text nepřevedl, aplikace text2wave by nečitelné znaky nepřečetla, nebo přečetla neprávně a vznikl by tak nečitelný audio soubor. Po převodu se text uloží do textového souboru, ten je pak již zmíněnou text2wave aplikací převeden na řeč uloženou ve zvukovém souboru. [27]

Aby mohlo Voiceglue zpracovávat VoiceXML dokumenty předané z Asterisku, je nutno vytvořit příkaz v číslovacím plánu, který bude odkazovat na příslušný VoiceXML dokument. Jedná se o příkaz AGI (Asterisk Gateway Interface), konkrétně FastAGI, který komunikuje se serverem pomocí síťového spojení. V mém případě budou všechny zpracovávané dokumenty uloženy na stejném lokálním serveru, ale v praxi tomu tak být nemusí, a pro každou aplikaci může být VoiceXML dokument uložený pokaždé na odlišném serveru. Nastavení URL adresy VoiceXML dokumentu můžeme nastavit staticky v souboru */etc/voiceglue.conf*, který obsahuje i nastavení adresáře obsahujícího zvukové soubory Asterisku. Pokud tedy provedeme statické nastavení cesty k dokumentu, budeme pak volat aplikaci AGI příkazem *Agi(agi://localhost)*. Tímto se ale omezíme jen na jednu spustitelnou VXML aplikaci. Pokud chceme zadávat URL adresu dokumentu přímo do

číslovacího plánu, pak nastavení */etc/voiceglue.conf*, neprovádíme a příkaz pro spuštění aplikace pak vypadá následovně:

```
Agi(agi://localhost/url=http%3A%2F%2Flocalhost%2Fapp1%2Froot.vxml)
```

Odkaz na VoiceXML dokument je zadán v URL kódování popsané v dokumentu RFC 3986, který definuje rezervové znaky, které se kódují pomocí znaku procenta, po kterém následuje hexadecimální ASCII kód znaku.

Tabulka 5.2: Rezervované znaky URL kódování

znak	!	#	$	&	'	(	)	*	+	,	/	:	=	?	@	[	]
kód	%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%5B	%5D

6 Úloha 1 – Dopravní informace

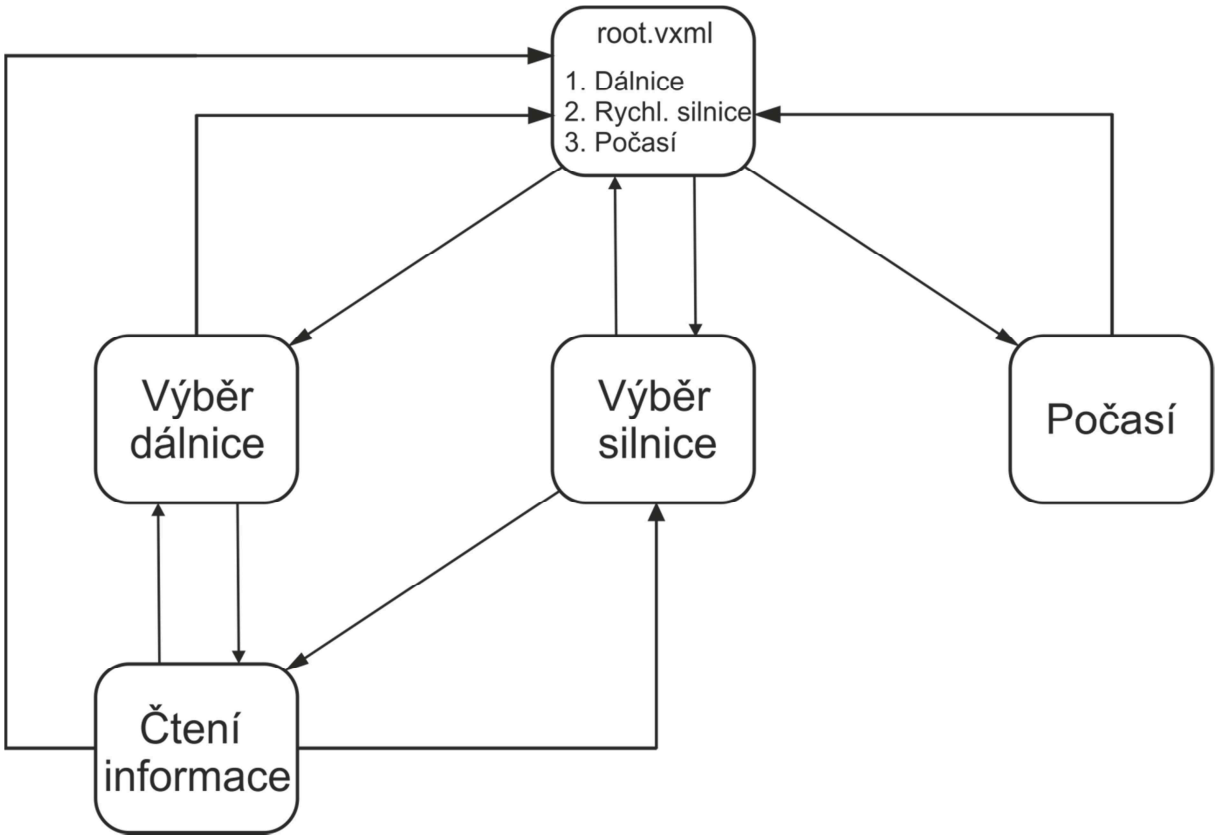
6.1 Návrh řešení

Cílem aplikace je poskytnout uživateli informace o stavu českých silnic a dálnic a také poskytnout předpověď počasí na týden dopředu. Aplikace obsahuje nabídku nejfrekventovanějších silničních tahů v České Republice a informace o nich sbírá z RSS čtečky webu Dopravní info.cz. Informace o počasí jsou taktéž poskytovány RSS kanálem, a to ze zpravodajského serveru ČTK České noviny.cz.

Aplikace obsahuje hlavní menu, kde si uživatel vybere oblast:

- Dálnice
- Rychlostní silnice
- Předpověď počasí

Po hlavním výběru informací z dálnic nebo rychlostních silnic si musí uživatel vybrat jednu položku ze seznamu dálnic nebo rychlostních silnic, o které chce slyšet informace, viz obrázek 6.1.



Obrázek 6.1: Diagram úlohy Dopravní informace

Po přečtení vybrané informace je uživatel opět vybídnut k výběru, má několik možností:

- Vrátit se do hlavní nabídky
- Vrátit se k výběru jiného názvu rychlostní komunikace
- Skočit do nabídky jiného druhu rychlostní komunikace
- Ukončit hovor

Pokud uživatel zvolí v hlavní nabídce předpověď počasí, bude informován o stavu počasí na celý další týden, poté bude automaticky vrácen zpět do hlavní nabídky.

6.2 Praktické řešení

6.2.1 Hlavní nabídka

Kořenový dokument aplikace *root.vxml* je psán jazykem VoiceXML. Jeho kompletní zdrojový kód je uveden příloze A. Soubor obsahuje základní blok, kde se nachází text s uvítáním uživatele a oznámení kam se dovolal. Po něm aplikace předá dialog, pomocí značky <*goto*>, do menu s názvem *hlavniMenu*. Zde se nachází hlavní nabídka, ze které si uživatel volí typ rychlostní komunikace. Jeho volba je pak rozpoznávána pomocí značky <*choice*> s atributem *next*, který definuje, kam se má dialog předat. Celá značka vypadá následovně:

```
<choice dtmf="1" next="dalnice.vxml"/>
```

V tomto případě se po stisknutí tlačítka 1 přesune dialog do dokumentu s názvem *dalnice.vxml*. Po definici vstupu jsou pak v dokumentu ještě obsaženy značky, které hlídají správnost vstupu, takže pokud uživatel zadá číslo mimo rozsah nabídky, bude na to upozorněn a požádán o zadání nového vstupu. Pokud by se uživatel rozhodl hovor ukončit, může tak učinit zavěšením, nebo stisknutím příslušného tlačítka. V tom případě pak dialog převezme formulář s názvem *konec*, který se s uživatelem se rozloučí a aplikaci ukončí.

6.2.2 Druhá nabídka

Jedná se o soubory *dalnice.vxml* a *rychlostni.vxml*, které jsou součástí přílohy této práce, jedná se konkrétně o přílohu B a přílohu C.

Do druhé nabídky se uživatel dostane, pokud si v hlavní nabídce vybere, že by chtěl slyšet informace z dálnic nebo z rychlostních silnic. Dokument můžeme rozdělit do dvou důležitých částí. V první části se nachází seznam názvů nefrekventovanějších rychlostních komunikací v České Republice, ze kterých si uživatel může již konkrétně vybrat. Tímto jsme se dostali k druhé části, která obsahuje značky s odkazy na webový frontend, který vygeneruje dané zprávy. Jedná se o PHP kód, jehož funkci popíši v kapitole 6.2.3

V souboru *dalnice.vxml* je druhá část odlišná od části v souboru *rychlostni.vxml*. Definice volby uživatelského vstupu jsou zde provedeny jinak. Je to z důvodu delší prodlevy zpracování informací u některých dálnic. Typicky dálnice D1 nebo dálnice D3 či D5, u kterých je množství informaci daleko větší než u jiných rychlostních komunikací. Při zpracovávání těchto informací docházelo prodlevě až deseti vteřin, to mělo za následek dlouhou pauzu v hovoru, kterou bylo potřeba eliminovat. Rozhodl jsem se tedy, že během zpracovávání bude hrát na pozadí hudba a bude přečtena hláška „Čekejte prosím, Váš požadavek se zpracovává." Musel jsem tedy nejdříve vytvořit zvukový soubor s tímto obsahem, což jsem provedl tak, že jsem vzal zvukový soubor, který se používá k tzv. „Music On Hold" z aplikace Asterisk a k němu jsem přidal výstupní zvukový soubor z TTS Festival, který převedl požadovanou hlášku do řeči. Vložit tento soubor do aplikace VoiceXML je možný podle dokumentace hned několika způsoby. [28] [29] Ovšem vzhledem k tomu, že VoiceGlue zatím plně nepodporuje všechny značky a funkce VoiceXML 2.1, vložení souboru do parametru *fetchaudio* značky *choice* nefunguje. Musel jsem tedy použít samostatnou značku *audio*, která ale mezi značkami *choice* opět nefungovala. Byl jsem tedy nucen rozlišovat DTMF vstup od uživatele pomocí podmínkových značek *if*, *elseif* a *else*, a musel jsem také nadefinovat gramatiku, kterou jsem pro rozpoznávání vstupu používal. Testování vstupu pak vypadalo následovně:

```
<if cond="vyber == '1'">
<audio src="hlaska.wav"  fetchhint="prefetch"/>
<goto next="doprava.php?nazev=D1" />
```

Dokument musí obsahovat pojmenovanou značku *field*, která se dává dovnitř formuláře, tyto značky zajišťují snadnější shromažďování informací od uživatele, ty jsou pak porovnávány s aktivními gramatikami. Formulář proto musí obsahovat definici gramatiky nebo odkaz na ni. Tato gramatika udává, jakých hodnot může uživatelův vstup nabývat, v našem případě jsou to tedy všechny číslice. Vyhodnocení podmínek probíhá mimo shromažďovací pole, slouží k tomu přímo elementy *filled*. Jestliže je podmínka splněna, načte se zvukový soubor, a provede se předání dialogu, pokud ovšem načtení skriptu trvá, přehraje se zvukový soubor, aby bylo vyplněno ticho během zpracovávání. Značka *audio* s parametry *src*, který odkazuje na zvukový soubor, dále je uveden parametr *fetchhint* ten může nabývat dvou hodnot:

- **prefetch** – zvukový soubor bude načten předběžně
- **safe** – zvukový soubor bude načten pouze na výzvu aplikace, nikdy dříve

6.2.3 Základ skriptu pro zpracování informací

Pro získávání informací jsem použil RSS kanál z webových serverů. Výstup čtečky může vypadat následovně:

```
<rss version="0.91">
  <channel>
  <title>ČN: Předpověď počasí</title>
  <link>http://www.ceskenoviny.cz/pocasi/</link>
  <description>ČN: Předpověď počasí</description>
  <language>cs</language>
  <copyright>ČTK</copyright>
    <item>
      <title>Nadpis</title>
      <link>Odkaz na web</link>
      <description>Popis zprávy</description>
    </item>
  </channel>
</rss>
```

Důležité informace pro tuto aplikaci se nachází mezi elementy <*item*>, jedná se hlavně o data označená jako <*title*> a <*description*>. Abych tato data z výstupu čtečky získal, musel jsem použít nějaký program nebo skript, který mi výstup rozparsuje. Rozhodl jsem se tedy použít PHP RSS čtečku od pana Daniela Tlacha. Jedná se o objektově orientovaný PHP kód, který dokáže zobrazit vybraná data získané z RSS kanálu. RSS Feed Reader verze 2.2 spadá pod licenci GNU GPL a je tak volně šiřitelná. [30]

Obsahem archivu čtečky jsou dva PHP soubory, *Rss.minified.php* a *Rss.php*, které obsahují definice tříd RSS čtečky. Soubor *Rss.minified.php* musí být vložen do PHP skriptu zpracovávajícího potřebný RSS kanál. Oba soubory pak musí být ve stejné složce jako samotný vytvořený PHP skript. Volání RSS čtečky pak probíhá následovně:

```
$RSS = new Rss;
$feed = $RSS->getFeed($odkaz, Rss::XML);
```

Proměnná *$odkaz* obsahuje odkaz na RSS kanál, ten je čtečkou otevřen a jednotlivá data z kanálu jsou pak uložena do pole *$feed*, ke kterým pak přistupujeme např. pomocí cyklu, který pole projde.

Získání užitečných dat pak můžeme docílit následovně:

```
foreach($feed as $item)    {
      echo "<prompt>".$item["title"]."</prompt>\n";
      echo "<prompt>".$item["description"]."</prompt>\n";
      }
```

Na lichých řádcích potom budeme mít nadpisy událostí a na sudých budou jejich popisy.

Na prvních deseti řádcích mého skriptu je deklarace XML dokumentu, která zajistí správné formátování výstupního VoiceXML souboru. Dále je zde obsaženo vložení kódu výše zmíněné RSS čtečky. Dalším poměrně zákeřným problémem, na který jsem během zpracovávání diplomové práce narazil, byla omezená délka textu, kterou je schopna aplikace VoiceGlue předat správci řečových syntéz k převodu na řeč. Maximální délka textu se pohybovala okolo 230 znaků. Textový řetězec mezi značkami <*prompt*>, ve VoiceXML dokumentu, který byl delší, se jednoduše přeskočil a nedošlo vůbec k jeho přečtení. Tudíž bylo zapotřebí takovýto textový řetězec rozdělit na menší celky ohraničené značkami <*prompt*>. Rozhodl jsem se tedy, že budu textové řetězce dělit po větách, kdy oddělovací znak bude tečka, vykřičník nebo otazník následovaný mezerou. V případě některých dopravních zpráv ze silnic a dálnic, kdy jsou tyto zprávy psány stručně a často i heslovitě jsem byl nucen přidat jako oddělovací znak ještě středník. Pro rozdělení textu na jednotlivé věty jsem použil PHP funkci *preg_split()*, která využívá regulárních výrazů:

```
$vety=preg_split('/(?<=[.?!,])\s+/', $text, -1,PREG_SPLIT_NO_EMPTY);
```

Funkce vytvoří pole, jehož jednotlivé prvky jsou části proměnné *$text*, rozdělené podle regulárního výrazu v prvním argumentu. Třetí argument určuje limit, tzn. maximální počet prvků pole. Pokud je limit nastaven na -1, pak je limit neomezený. Nastavením posledního argumentu na *PREG_SPLIT_NO_EMPTY* nám zajistí, že funkce nevrátí prázdný výsledek.

Dalším nepříjemným zjištěním bylo, že pokud se v textu nachází uvozovky, je tato část textu opět při zpracovávání aplikací VoiceGlue přeskočena, naštěstí je řešení snadné. Uvozovky a další znaky, které nechceme interpretovat, jednoduše při procházení textu odstraníme.

Tento segment skriptu, obsahující deklaraci XML dokumentu, vložení RSS čtečky a dělení textových řetězců, je jako základ obsažen ve všech PHP skriptech použitých v této práci.

6.2.4 Generování dokumentu s informacemi ze silnic a dálnic

PHP skript zpracovávající informace z českých silnic a dálnic je přiložen jako příloha D této diplomové práce. Ve zprávách o dopravní situaci na dálnicích a silnicích jsou plná zkratek a odporných pojmů. Když tyto informace čteme, tak zkratky nečteme jako zkratky, ale v jejich plném významu. VoiceGlue interpret ale tyto zkratky nezná a proto je zapotřebí mu text podat tak, aby mu

posluchač rozuměl. Musíme tedy zajistit jejich převod na plný význam. Problém jsem řešil tak, že jsem zkratky a podobné informace, kterým by bylo při klasickém přečtení hůře rozumět, vyhledával a nahrazoval již při zpracovávání PHP skriptem. Použil jsem k tomu PHP funkci *str_replace()*, která projde textový řetězec, a zadané podřetězce nahradí jinými podřetězci. Výhodou této funkce je, že dokáže pracovat s poli, tzn., že znak nebo řetězec, který je uložen v prvku pole na určité pozici, je nahrazen prvkem na stejné pozici jiného pole. Čtení zkratek pak může vypadat následovně:

```
$zkratky = array ('m/s','PČR','BUS',);
$vyznam = array ('metrů za sekundu','Policie','autobus');
$vystup = str_replace($zkratky, $vyznam, $vstup);
```

Máme deklarováno jedno pole se zkratkami a druhé s jejich významem, důležité je, aby odpovídali jejich jednotlivé pozice, jinak dojde k přehození významu. Funkce prochází vstupní text *$vstup*, a pokud narazí na podřetězec, který je shodný s jakýmkoli prvkem pole *$zkratky*, nahradí jej prvek stejné pozice z pole *$vyznam*.

Pro vytvoření VoiceXML dokumentu s korektními informacemi bylo zapotřebí předat skriptu výsledek uživatelovi volby. Inspiroval jsem se webovými stránkami a zadal jsem jej pomocí metody GET, kdy se zadaná data posílají jako součást URL za otazníkem. Vytvořil jsem PHP funkci s názvem *getClanky*, která implementuje řešení všech výše zmíněných problémů, a jako argument funkce jsem zvolil výsledek uživatelovi volby, tedy název rychlostní komunikace. Výstupem funkce jsou pak informace z RSS kanálu, které obsahují v nadpisu události název zadané rychlostní komunikace. Poslední statická část skriptu obsahuje VoiceXML dokument, který implementuje výsledek funkce *getClanky* a vytváří další nabídku pro předání nebo ukončení dialogu.

6.2.5 Generování dokumentu s předpovědí počasí

PHP skript zpracovávající informace o předpovědi počasí je přiložen jako příloha E této diplomové práce. Struktura výsledného dokumentu je celkem jednoduchá. Obsahuje blok, ve kterém jsou informace o předpovědi počasí na týden dopředu. Zkratky a jednotky veličin obsažené v textu jsou nahrazeny pomocí PHP funkce *str_replace* jejich významy. Jediný problém, na který jsem při vyváření této části aplikace narazil, byl neschopnost správce řečových syntéz přečíst datum. Vzhledem k tomu, že nejsem expert v PHP programování, nedokázal jsem tento problém efektivně vyřešit. Vzhledem k tomu, že se datum vyskytuje pouze v nadpisech prvních dvou následujících dní, rozhodl jsem se, že datum z aplikace vypustím. Bylo tedy potřeba první dva nadpisy rozdělit tak, aby se datum dalo smazat. Nadpisy mají vždy stejný formát, jedná se o 4 řetězce oddělené mezerou, kdy poslední řetězec je vždy datum. Takže jsem pomocí PHP funkce *explode()* rozdělil tento nadpis a zpětně jej opět složil, ovšem již bez data. Provedl jsem to následovně:

```
$i=1;
foreach($feed as $item)    {
  if($i<3){
   $title = explode(" ", $item["title"]);
   echo "<prompt>"$title[0]." ".$title[1]." ".$title[2]."</prompt>";
  }
  else  echo = "<prompt>".$item["title"]."</prompt>";
 $i++;
}
```

RSS kanál o počasí má celkem tři nadpisy, ale datum se objevuje pouze u prvních dvou. Proto bylo potřeba pohlídat, aby se třetí nadpis nedělil, ale zůstal celý. To jsem dosáhl díky proměnné *$i*, které jsem dal na začátku skriptu hodnotu 1, a postupně ji navyšoval po každém kole cyklu. Pokud byla proměnná *$i* menší jak 3, což znamenalo, že cyklus ještě neprošel dvakrát, vyhověla podmínka a provedlo se rozdělení nadpisu, jakmile cyklus proběl podruhé, proměnná *$i* dosáhla hodnoty 3, podmínka již nevyhovuje a rozdělní se neprovádí.

Ve výsledném dokumentu je po vygenerování informace o předpovědi počasí přidána značka <*goto*> zajišťující automatické předání dialogu zpět do hlavní nabídky aplikace.

6.2.6 Nastavení spuštění aplikace v Asterisku

Posledním krokem před uvedení aplikace do chodu je potřeba zavést odkaz na kořenový dokument aplikace do číslovacího plánu Asterisku. Upravíme tedy soubor *extensions.conf* v adresáři */etc/asterisk*, a přidáme následující řádky:

```
exten => 101,1,Answer
exten => 101,n, \
Agi(agi://localhost/url=http%3A%2F%2Flocalhost%2Fapp1%2Froot.vxml)
exten => 101,n,Hangup
```

Nyní je potřeba se přihlásit do konzole Asterisku, provést *reload* a pak ještě provést restart VoiceGlue. Potom se po zavolání na číslo 101 spustí aplikace Dopravní informace.

7 Úloha 2 – Čtení zpráv nevidomým

7.1 Návrh řešení

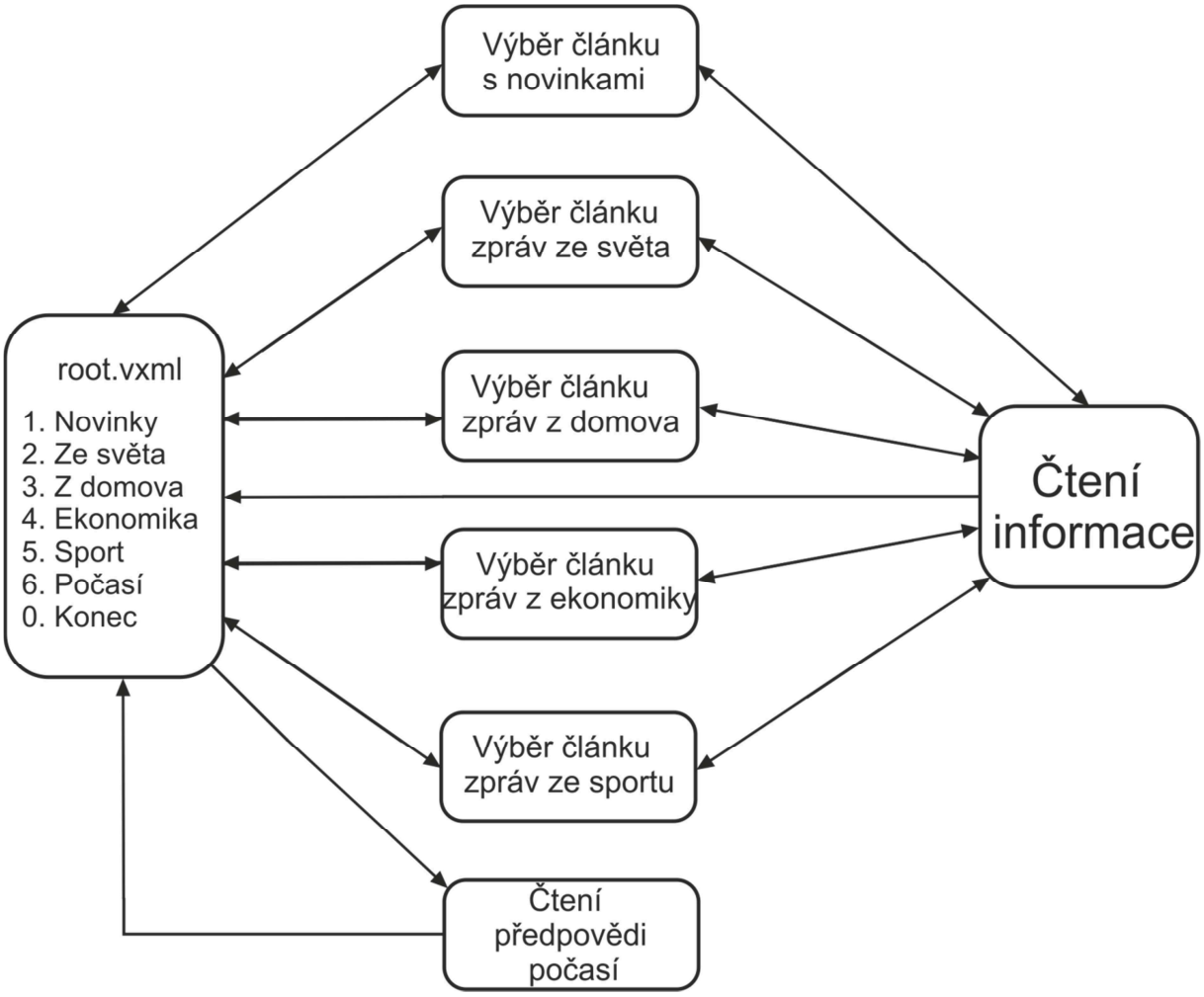
Cílem úlohy je informovat zrakově postižené lidi o každodenním dění čtením zpráv ze serveru Novinky.cz a předpovědi počasí to ze zpravodajského serveru ČTK České noviny.cz. Čtené informace budou vybírány z následujících kategorií:

- Nejnovější zprávy
- Zprávy ze světa
- Zprávy z domova
- Zprávy ze světa ekonomiky
- Sportovní události
- Předpověď počasí

Část aplikace, ve které se uživatel dozvídá předpověď počasí, je totožná s části o počasí z předchozí úlohy. Po výběru kategorie bude uživateli přehrána nabídka devíti nejnovějších článků z této skupiny, ze kterých si bude moci vybrat, nebo se bude moct vrátit zpět do hlavní nabídky a zvolit jinou kategorii. Pokud si zvolí článek, bude mu následně přečten. Potom se může vydat některou z následujících cest:

- Vybrat jiný článek ze stejné skupiny
- Návrat do hlavní nabídky
- Ukončit hovor

Celý diagram aplikace znázorňuje obrázek 7.1



Obrázek 7.1: Diagram úlohy Čtení zpráv nevidomým

7.2 Praktické řešení

7.2.1 Hlavní nabídka

Zdrojový kód hlavní nabídky aplikace je kompletně uveden v příloze F. Jedná se o VoiceXML dokument s úvodním uvítacím blokem a následnou nabídkou kategorií. Výběr kategorií probíhá způsobem popsaným výše, v kapitole 6.2.1. Celá struktura dokumentu je stejná jako hlavní nabídka úlohy s dopravními informacemi.

7.3 Výběr a následné čtení článků

Nejjednodušší řešení získávání informací bylo použití RSS kanálu, pro jehož čtení jsem použil stejnou PHP RSS čtečku, jako jsem použil v aplikaci s dopravními informacemi. Zdrojový kód skriptu, který řeší výběr a čtení článků je přiložen jako součást práce v příloze G.

7.3.1 Struktura webového informačního portálu

Před tím, než jsem se pustil do vytváření skriptu, bylo potřeba prostudovat struktury informačních webů a jejich RSS kanály. Hlavním požadavkem, podle kterých jsem web vybíral, byl přijatelný a dobře zpracovatelný výstup RSS kanálu. Dalším neméně důležitým kritériem bylo samotné zpracování webové stránky s konkrétní zprávou, hlavně její uspořádání zdrojového kódu. Všem těmto kritériím vyhověly weby Novinky.cz a iDnes.cz, ze kterých jsem posléze vybral k dalšímu zpracování aplikace server Novinky.cz, který měl výstup RSS kanálu, dle mého názoru, lépe strukturován.

Portál Novinky.cz má pro každou vybranou kategorii vlastní RSS kanál, kde jsou uvedeny titulky jednotlivých článků, odkazy na webovou stránku s tímto článkem, a pak popis článku, který je shodný s prvním odstavcem pod hlavním titulkem článku na webové stránce. Webová stránka je pak dále rozdělena fotografií či obrázkem nebo videem, po kterém pak následuje zbylý text článku, ve kterém se pak mohou nacházet další obrázky nebo videa. Tento oddíl webu je ohraničen značkami <*div*> s identifikátorem *articleBody* u všech vybraných kategorií kromě kategorie *Sport*, ve kterém je oddíl pojmenován jako *article-text*.

7.3.2 Zpracování RSS kanálu

Předání volby kategorie se prování metodou GET, kdy je tato volba součástí URL adresy volání skriptu. K jednotlivé kategorii je potřeba přiřadit správný odkaz na RSS kanál, což je provedeno jednoduchým testováním argumentu *zprava*. RSS kanál je zpracováván pomocí RSS čtečky popsané v kapitole 6.2.3. Jediným rozdílem ve volání RSS čtečky je v druhém argumentu funkce *getFeed*, ten je nastaven na *Rss::TXT*. Takto nastavený argument zajistí použití regulárních výrazů pouze pro textový výstup. Ovšem i takto upravený výstup obsahuje v popisu článku jeden obrázek a vždy u prvního článku se v jeho popisu objevují i oddíly s reklamou.

Struktura VoiceXML dokumentu zajišťující výsledný výběr článků a jejich čtení lze rozdělit do několika částí. V první části se nachází nabídka článku obsahující jejich hlavní nadpisy. Tuto část dokumentu generuje funkce *getRSSNadpisy*, která prování naplnění RSS čtečky odkazem na vybraný RSS kanál. Výstupem této funkce je seznam devíti nejnovějších článků dané kategorie, které jsou pak seřazeny do nabídky. Další částí dokumentu je definice gramatiky, kde jsou definovány jako platné vstupy všechny číslice, a ošetření platnosti zadání vstupu. Poslední a také největší částí je samotné rozpoznání vstupu s obsahem vybraného článku. Rozpoznání vstupů se prování pomocí značek <*if*>, <*elseif*> a <*else*>. Pokud je splněna podmínka uvedená v těchto značkách, je provedeno čtení textu mezi značkami <*prompt*>.

7.3.3 Získávání informací z webové stránky

Informace z webové stránky jsou pro tuto aplikaci velmi zásadní. RSS kanál obsahuje pouze titulek a stručný popis události, celý článek je uveden až na webové stránce, na kterou odkazuje položka <*link*> v RSS kanálu. Vzhledem k tomu, že má webová stránka serveru Novinky.cz celkem jednoznačnou členitost a téměř stejně pojmenované oddíly s plnohodnotnými informacemi, rozhodl jsem se data z webu získat pomocí PHP, konktrétně pomocí třídy *DOMDocument*.

DOM (Document Object Model) je standardní rozhraní pro práci s dokumenty XML definované konsorciem W3C. Toto rozhraní definuje způsob, jakým se dokument XML mapuje na hierarchii objektů v paměti. Každé části dokumentu, jako je element, atribut, textový uzel apod. odpovídá v paměti jeden objekt. Pomocí metod a vlastností dostupných na každém objektu můžeme zjišťovat druh uzlu, jaký ve stromu dokumentu XML zastupují, jejich název, obsah, seznam objektů reprezentujících dětské uzly, objekt zastupující rodiče uzlu atd. Dokument je v paměti reprezentován jako objekt, který je instancí třídy *DOMDocument.* Pomocí metody *load()* je možné vytvořit DOM reprezentaci dokumetu XML uloženého v souboru. Třída dokáže pracovat s HTML soubory stejně jako s XML soubory. [31] [32]

Načtení HTML stránky článku z RSS kanálu do paměti lze provést následovně:

```
$doc = new DOMDocument('1.0', 'utf-8');
$doc->loadHTMLFile($item["link"]);
```

Ze zdrojového kódu webové stránky jsem zjistil, že požadovaný text je ohraničen oddílovou značkou <*div*> s elementem *id=“articleBody“*, v případě sportovních událostí se jedná o element *id=“article-text“*. Pro získání textového obsahu tohoto oddílu jsem použil funkci *getElementById*, jejíž argumentem je název elementu *id*. Text k přečtení se tedy skládá ze stručného popisu článku a textu získaného touto funkcí. Problémem je ovšem to, že informace jsou primárně určeny pro webovou prezentaci a obsahují různé objekty, jako jsou např. obrázky, videa, tabulky, reklamy apod., které jsou pro VoiceXML dokument této aplikace nevhodné. Je tedy zapotřebí výsledný získaný text projít a tyto objekty odstranit. Problém jsem ve třech krocích.

V prvním kroku jsem pomocí regulárních výrazů odstraňoval z popisu článku obrázky a odkazy na jiné webové stránky. Požil jsem k tomu PHP funkci *preg_replace*:

```
preg_replace("~(<img|<a href)[^>]+\>~i", "", $item["description"]);
```

Tato funkce vyhledá řetězec, který odpovídá zadanému regulárnímu výrazu v prvním argumentu a nahradí jej řetězcem nebo znakem v argumentu druhém. V tomto případě tedy smaže všechny řetězce, které začínají znaky <*img* nebo <*a href* a končí znakem >.

V druhém kroku provádím čištění textového obsahu z webové stránky, které obsahuje spousty objektů, které nejsou vhodné pro interpretaci řečí. Vytvořil jsem si k tomuto účelu funkci *vycistiText*, která za úkol vyhledávat zadané názvy značek a mazat nebo upravovat jejich obsahy. Funkce vyhledává následující značky:

- Table – tabulka, jejíž přítomnost v naší aplikaci je zbytečná
- Script – obsahem je video, které nelze do tohoto VoiceXML dokmentu začlenit
- p – odstavec, na jehož konec je přidána mezera pro lepší dělení textu

Funkce dále vyhledává a maže obsah značek, jejichž atribut *class* nabývá následujících hodnot:

- acmDescription – popis vloženého obrázku
- acmAuthor – jméno autora obrázku
- blind – popis vloženého videa

Vrácený text je v podstatě ten nejdůležitější text, který každý uživatel webové stránky s článkem přečte. Neobsahuje již žádné části, které slouží k vizuální prezentaci informace.

V posledním kroku provádím čištění textu od nežádoucích znaků, které vedou k chybě VoiceXML interpreta, nebo ke špatnému porozumění čteného textu. Používám k tomu PHP funkci *str_replace*. Po těchto úpravách textu pak dochází k dělení textu po větách a dalším úpravám výsledného VoiceXML dokumentu, které jsou popsány v předchozí kapitole, tak aby byl korektně interpretován.

7.3.4 Nastavení spuštění aplikace v Asterisku

Aby byla aplikace plně funkční, musí se přidat klapka do číslovacího plánu Asterisku s odkazem na kořenový soubor VoiceXML aplikace.

```
exten => 102,1,Answer
exten => 102,n, \
Agi(agi://localhost/url=http%3A%2F%2Flocalhost%2Fapp2%2Froot.vxml)
exten => 102,n,Hangup
```

Potom je nutné provést restart aplikace VoiceGlue, a znovu načíst nastavení ústředny příkazem *reload* v příkazové řádce Asterisku.

8 Závěr

Účelem této diplomové práce bylo implementovat aplikace VoiceXML do pobočkové ústředny Asterisk. Zpracovat teoretickou i praktickou problematiku nasazení VoiceXML dokumentů, které bude ústředna Asterisk využívat k nasazení širší koncepce dialogových systémů, jejichž textový výstup bude interpretován správcem řečových syntéz aplikací Text-to-Speech Festival.

V diplomové práci bylo nejprve obecně pojednáno o dialogových systémech a způsobech, kterými lze komunikovat s počítači a jinými přístroji. Byly rozebrány technologie rozpoznání a porozumění lidské řeči a následné popsání komponent dialogových systémů. Následující část práce obsahuje rozsáhlou studii o jazyku VoiceXML, byla popsána jeho evoluce vývoje, architektura jazyka i architektura aplikace postavené na tomto jazyku. Dále jsou pak představeny jednotlivé součásti aplikace, které byly použity pro praktickou část této diplomové práce. Bylo pojednáno o pobočkové ústředně Asterisk, kde jsem popsal možnosti jejího nasazení, podporu protokolů a kodeků. Další popisovanou součástí byl VoiceXML interpret, aplikace VoiceGlue, a správce řečových syntéz Festival.

Výsledkem této práce jsou dvě aplikace. První má za úkol informovat řidiče o dopravních situacích na českých sinicích a dálnicích a dále je informovat o předpovědi počasí na celý další týden. Informace sbírá data o situacích na rychlostních komunikacích z RSS kanálu webu Dopravní info.cz Data o počasí jsou interpretována ze zpravodajského serveru ČTK České noviny.cz. Druhá aplikace provádí čtení zpráv ze serveru Novinky.cz a předpověď počasí zrakově postiženým lidem. Obě aplikace jsou provedeny formou systému IVR pomocí VoiceXML dokumentů, kde si uživatel vybírá z dynamicky generovaných nabídek DTMF volbou.

Zadání práce se mi podařilo splnit, požadovaná implementace VoiceXML do pobočkové ústředny Asterisk byla realizována. Celkové řešení je open source, kde veškeré součásti implementace jsou vedeny pod licencí GNU GPL a tudíž lze výsledek této práce využít pro další řešení projektu vyvíjecího distribuci hlasových zpráv.

Použitá literatura

1. **KOPEČEK, Ivan.** Studijní materiály k předmětu Dialogové systémy. *Dialogové systémy.* [Online] 2009. http://www.fi.muni.cz/~kopecek/ds.htm.

2. **GÜZELDERE, Güven a FRANCHI, Stefano.** Colorful personalities. *http://www.stanford.edu.* [Online] Stanford University, 1995. http://www.stanford.edu/group/SHR/4-2/text/dialogues.html.

3. **PSUTKA, Josef a kol.** *Mluvíme s počítačem česky.* Praha : Academia, 2006.

4. **Vondra, Martin.** *Kepstrální analýza řečového signálu.* místo neznámé : Vysoké učení technické v Brně, 2001.

5. **W3C.** Voice Extensible Markup Language (VoiceXML) 3.0. [Online] 2010. http://www.w3.org/TR/voicexml30/.

6. **Forum, VoiceXML.** VoiceXML's History. [Online] 10. 03 2009. http://www.voicexml.org/voicexml-tutorials/introduction/voicexmls-history.

7. **W3C.** Voice eXtensible Markup Language (VoiceXML™) version 1.0. [Online] 2000. http://www.w3.org/TR/voicexml/.

8. —. Voice Extensible Markup Language (VoiceXML) Version 2.0. [Online] 2004. http://www.w3.org/TR/voicexml20/.

9. —. Voice Extensible Markup Language (VoiceXML) 2.1. [Online] 2007. http://www.w3.org/TR/voicexml21/.

10. **ROSS, Timothy.** *XML, Managing Data Exchange.* místo neznámé : Global Media, 2007. 8189940880.

11. **EDGAR, Bob.** *The Voicexml Handbook: Understanding and Building the Phone-Enabled Web.* místo neznámé : Elsevier Science Limited, 2001. 1578200849.

12. *Interactive VoiceXML Module into SIP-based Warning Distribution System.* **VOZŇÁK, Miroslav, a další.** 2011.

13. **McGlashan, S., Burnet, D. a Carter, J.** *Voice Extensible Markup Language (VoiceXML) Version 2.0.* místo neznámé : W3C Recommendation, 2004.

14. **W3C.** Voice Browser Call Control: CCXML Version 1.0. [Online] 2007. http://www.w3.org/TR/2007/WD-ccxml-20070119/.

15. —. State Chart XML (SCXML): State Machine Notation for Control Abstraction. [Online] 2007. http://www.w3.org/TR/2007/WD-scxml-20070221/.

16. **International, ECMA.** ECMAScript Internationalization API Specification. [Online] 2011. http://www.ecma-international.org/ecma-402/1.0/index.html.

17. **W3C.** Speech Synthesis Markup Language (SSML) Version 1.0. [Online] 2004. Speech Synthesis Markup Language (SSML) Version 1.0.

18. —. Speech Recognition Grammar Specification Version 1.0. [Online] 2004. http://www.w3.org/TR/speech-grammar/.

19. —. Semantic Interpretation for Speech Recognition (SISR) Version 1.0. [Online] 2007. http://www.w3.org/TR/semantic-interpretation/.

20. —. Pronunciation Lexicon Specification (PLS) Version 1.0. [Online] 2006. http://www.w3.org/TR/2006/WD-pronunciation-lexicon-20061026/.

21. **MADSEN, Leif, MEGGELEN, Jim Van a BRYANT, Russell.** *Asterisk: The Definitive Guide.* 2011.

22. **VOZŇÁK, Miroslav.** *Voice over IP.* Ostrava : VŠB - Technická univerzita Ostrava, 2009. 978-80-248-1828-3.

23. **Digium.** Asterisk Architecture. [Online] 2011. https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture,+The+Big+Picture.

24. **I6NET.** VoiceXML for every business. [Online] I6NET. http://www.i6net.com/.

25. **Campbell, Doug.** VoiceGlue. [Online] http://www.voiceglue.org/.

26. **GitHub.** VoiceGlue: Features and Architecture. [Online] https://github.com/voiceglue/voiceglue/wiki/Features-and-Architecture.

27. —. Voiceglue 0.14 installation instructions. [Online] https://github.com/voiceglue/voiceglue/wiki/Voiceglue-0.14-installation-instructions.

28. **Voxeo.** Element <audio>, VoiceXML. *VoiceXML 2.1 Development Guide.* [Online] http://www.vxml.org/frame.jsp?page=audio.htm.

29. —. Element <choice>, VoiceXML. *VoiceXML 2.1 Development Guide.* [Online] http://www.vxml.org/frame.jsp?page=choice.htm.

30. PHP Classes. *RSS Feed Reader.* [Online] http://www.phpclasses.org/package/3724-PHP-Parse-and-display-items-of-an-RSS-feed.html#files.

31. **W3C.** Document Object Model (DOM). [Online] http://www.w3.org/DOM/.

32. **Schools, W3.** HTML DOM Tutorial. [Online] http://www.w3schools.com/htmldom/.

33. **Vozňák, Miroslav, Řezáč, Filip a Zdrálek, Jaroslav.** *Danger Alert Communication System.* Rio de Janeiro : autor neznámý, 2010. ISBN 978-85-228-0565-5.